THE UNIVERSITY OF NEW MEXICO

ECE 538
Advanced Computer Architecture

Instructor: Lei Yang

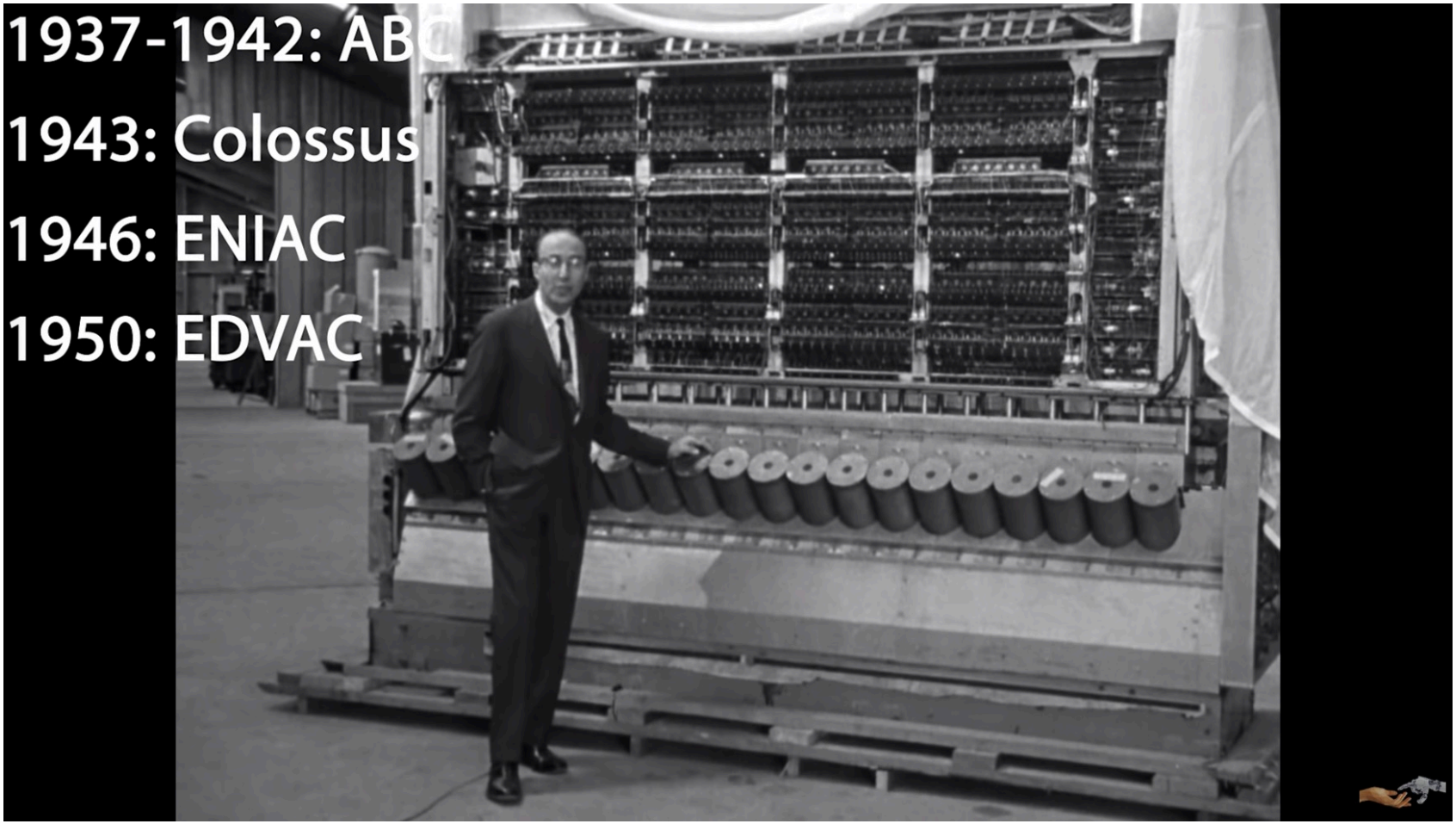
Department of Electrical and Computer Engineering

August 22, 2021

Introduction of
Computing and Computer Architecture

- Most materials can be referred to chapter 1
- The overview about the computer architecture
- Computer abstractions and techniques

The History of Computing

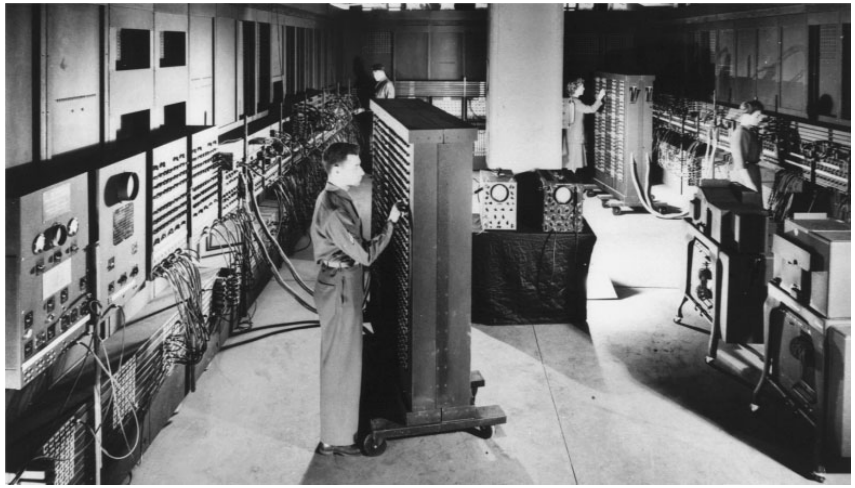


(Click figure to watch the video)

Computer Devices

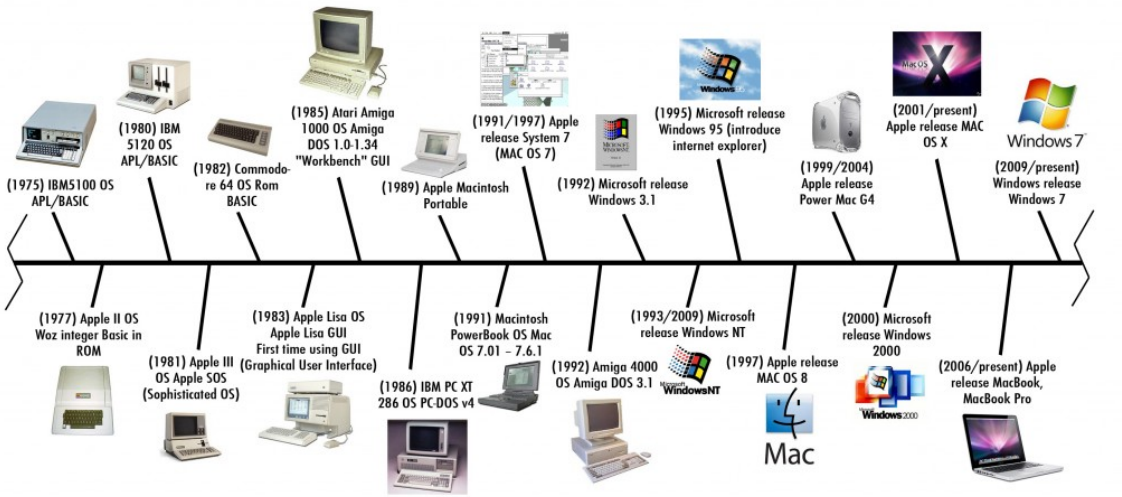
- ENIAC (Electronic Numerical Integrator and Computer)

  The world's first general-purpose electronic computer. (1946)



- Computers are pervasive

The Computer Revolution



Agricultural Age

↓

Industrial Age

↓

Information Age

Computers lead to the third revolution in our civilization

Computing system	Mainframe	Mini computer	Personal computer	Embedded computer
Era	1950s on	1970s on	1980s on	2000s on
Form factor	Multi-cabinet	Multi-board	Single board	Single chip
Resource type	Corporate	Departmental	Family	Personal
Users/system	100s – 1000s	10s – 100s	1s	1/10s
Cost	$ 1 million +	$ 100Ks +	$1Ks – $10Ks	$1s – $100s
Total units	10Ks +	100Ks +	1 billions +	1 Trillions +

Great Ideas in Computer Architecture

- Design of Moore's Law (Gordon Moore (Intel))
  - (integrated circuits) Double every 18-24 months
  - "up and to the right"

Moore's Law

- Abstraction to Simplify Design
  - Make more productive
  - Simpler model at higher levels

Abstraction

- Make a Common Case Fast
  - Enhance performance better than optimizing the rare case
  - Experimentation and measurement

Common Case Fast

Great Ideas in Computer Architecture

- Performance via Parallelism/Pipelining/Prediction
  - Perform operations in parallel
  - Pipeline: a particular pattern of parallelism
  - Prediction: guess and start working than wait

- Hierarchy of Memories

cost of memory is majority of cost of modern machine

  - Top: faster and more expensive per bit the memory
  - Wider the base of the layer, the bigger the memory

- Dependability via Redundancy
  - Redundant components: take over when a failure occurs
  - Help detect failures

Computer Categories

- Super Computers
  - High-end scientific and engineering calculations
  - Highest capability but represent a small fraction of overall computer market
- Mainframe Computers
  - Huge, towering machines with lots of processing power
- Minicomputers
  - Smaller size
  - Like a less powerful mainframe computer
- Microcomputers
  - Cheaper, and practical
- Personal/Mobile Computers
  - General purpose, variety of software
  - Subject to the cost/performance tradeoff

……

Computer Categories

- Supercomputers — Flops (floating point operations per sec)
  - High-end scientific and engineering calculations
  - Highest capability but represent a small fraction of the overall computer market

- Mainstream Computers — IPS (instructions per sec)
  - Huge size
  - Towering machines with lots of processing power

A pair of IBM mainframes
(Left is IBM Systems z13. Right is the IBM LinuxONE)

Computer Categories

- Microcomputers
  - Small size
  - Relatively expensive
  - Powerful computing

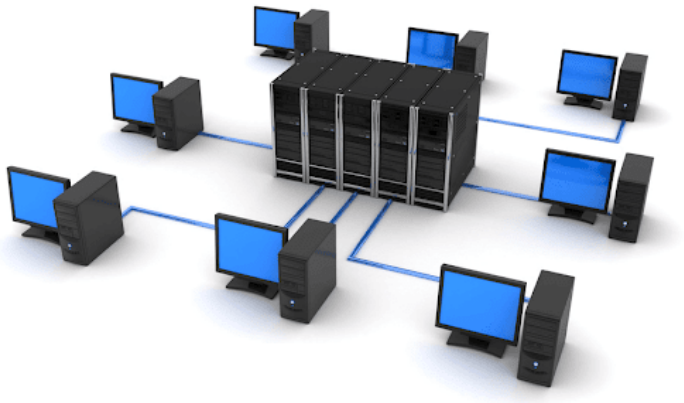
Raspberry Pi
Popular microcomputer

PMD (Personal Mobile Device)

- Mobile/Personal Computers
  - E.g. start phones, tablet computers
  - For general purpose
  - Variety of software versions
  - Emphasis on energy efficiency and real-time
  - responsiveness and are key for multimedia

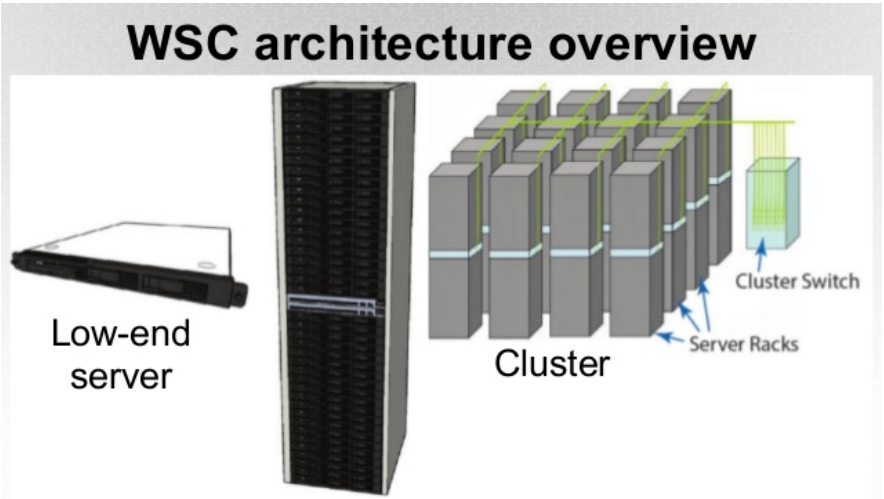
Computer Categories

client - server model



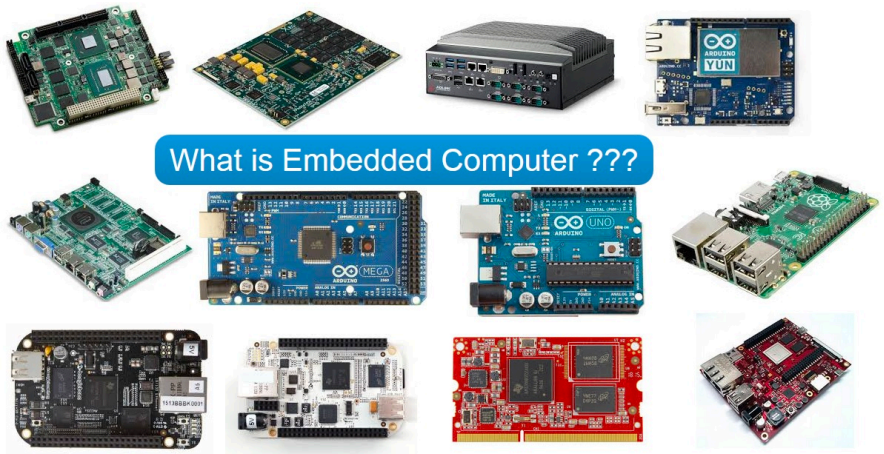
- Server Computers
  - Network based
  - Range from small servers to building sized
  - Emphasis on availability, scalability, throughput
  - replacing the traditional mainframe
  - mail, db, web servers
  - QoE vs. QoS – actual results matter more than raw power
- Cluster/Warehouse-Scale Computers (WSC)
  - Used for "Software as a Service (SaaS)"
  - Class of tens of thousands of servers
  - Rely on the software layer
  - Emphasis on Availability, Price-performance, Scalable, Distributed, Cost-efficiency

  (Sub-class: supercomputers)



Computer Categories

- Embedded Computers
  - Hidden as the components of systems
  - Emphasis on the the Stringent constraints of system power, energy, application-specific performance, and cost
- Dedicated system functions vs. flexible are the difference between embedded and microcomputers



What you Will Learn

- The features of the computer architectures
- How programs are translated into the machine language
- How the hardware can execute the machine language
- The hardware/software interface
- What determines the program performance and how it can be improved
- How hardware designers improve performance
- What are multiprocessor systems and the parallel processing
- Emerging and customized computer architecture design
- …

The Content of This Course

- Performance calculations and benchmarking
- Power and energy constraints
- Instruction set architecture
- Exception handling and interrupt
- Pipelining, data forwarding and interlocking
- Memory hierarchy and caches
- Multiprocessors and cache coherency
- GPUs and advanced parallel processing
- …

What is Computer Architecture?

- 1950s to 1960s:

  Computer Architecture Course = Computer Arithmetic

- 1970s and 1980s:

  Computer Architecture Course = Instruction Set Design, especially

  ISA appropriate for compilers

- 1990s:

  Computer Architecture Course = Design of CPU, memory system, I/O system, Multiprocessors

- 2000's:

  Embedded computing, server farms, power management issues, network processors.

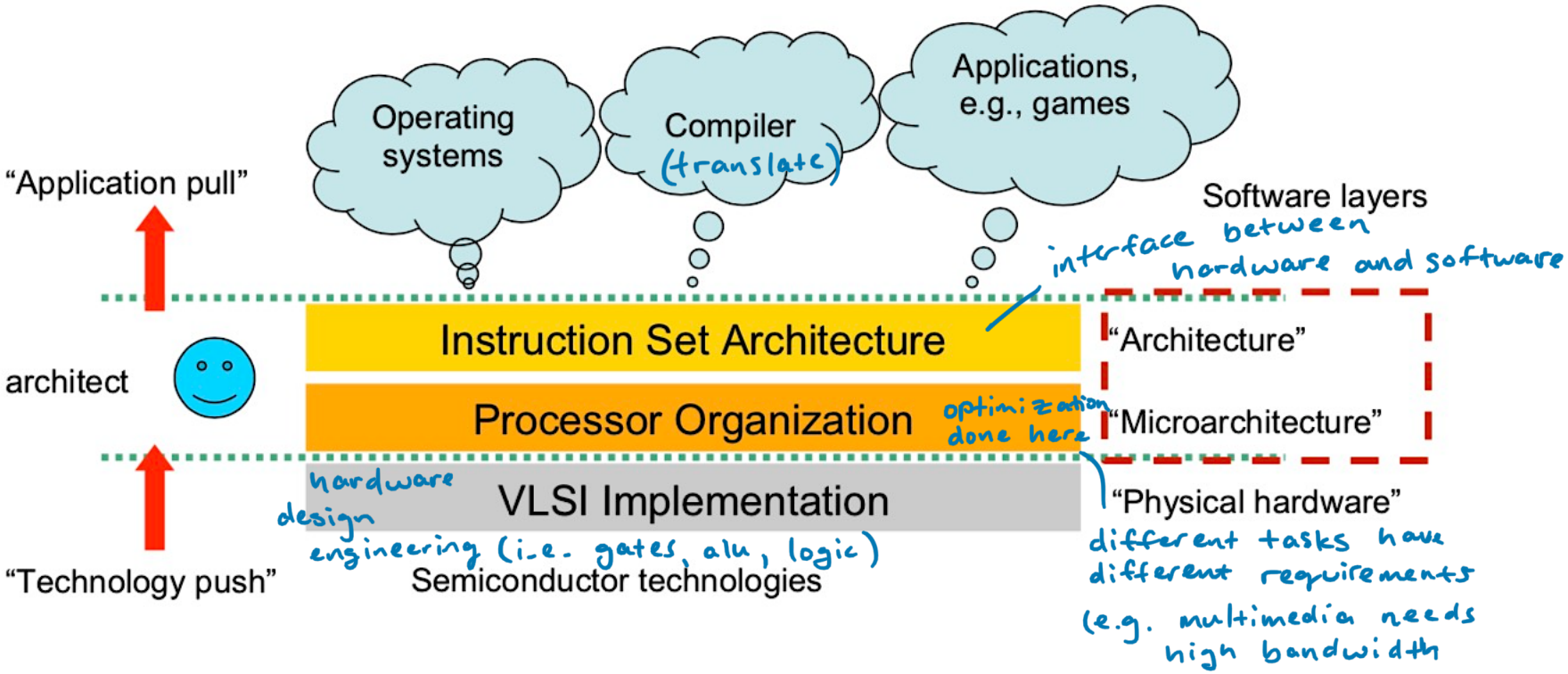
- 2012:

  All of the above + multicores + GPUs + cloud computing

What is Computer Architecture?

- Computer architecture deals with design and implementation of computer hardware. There are three main subcategories:
  - *Instruction Set Architecture, or ISA:* ISA defines the machine code that a processor reads and acts upon as well as the word size, memory address modes, processor registers, and data type.
  - *Microarchitecture, or computer organization:* describes implementation of the ISA. It specifies the execution of instruction through the control, storage and computing.
  - *System Design:* specifies all of the hardware components within a computing system, their functionalities and interconnection.

What is Computer Architecture?



- Models to overcome Instruction Level Parallelism (ILP) limitations:
  - Data-level parallelism(DLP)
  - Thread-level parallelism(TLP)
  - Request-level parallelism(RLP)

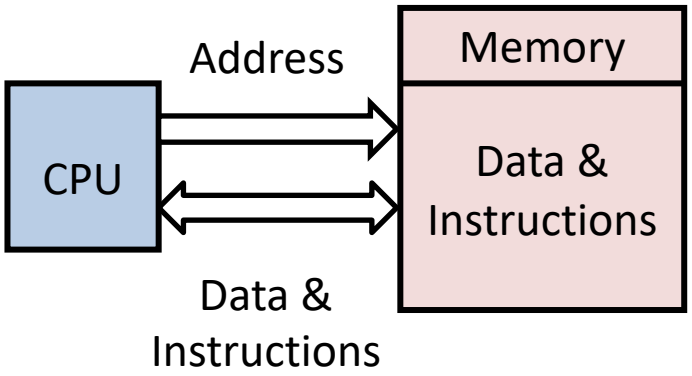
Review from last time

5 classes of computer devices

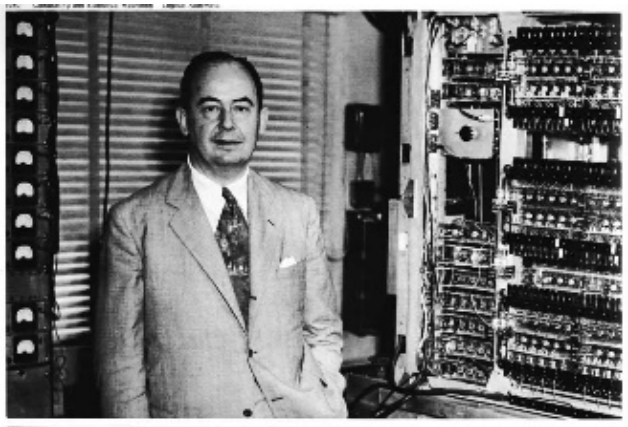
- super
- PMD
- servers
- embedded
- cluster/warehouse

Overview of Computer Systems I

- Von Neumann Architecture
  - Memory holds both data and instructions, both are transferred to the CPU through the same bus.



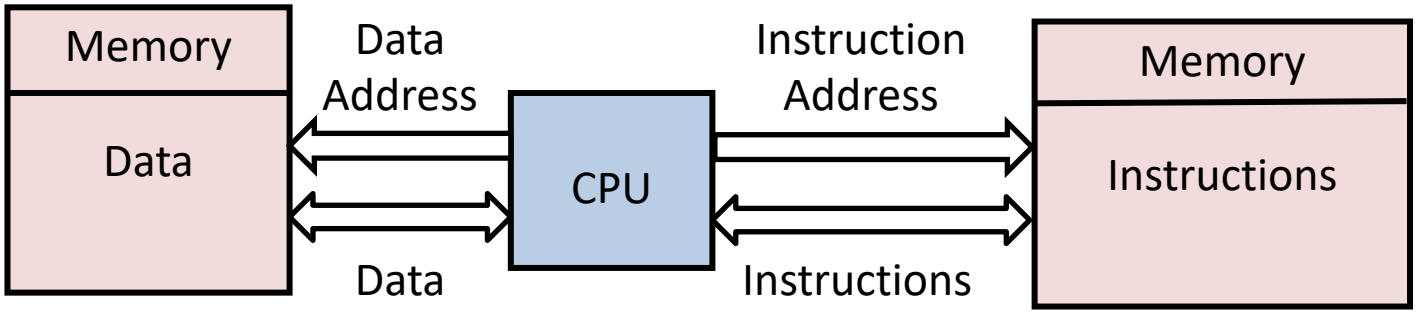
The von Neumann architecture



Von Neumann architecture can perform 1 memory access/clock cycle.

Overview of Computer Systems II

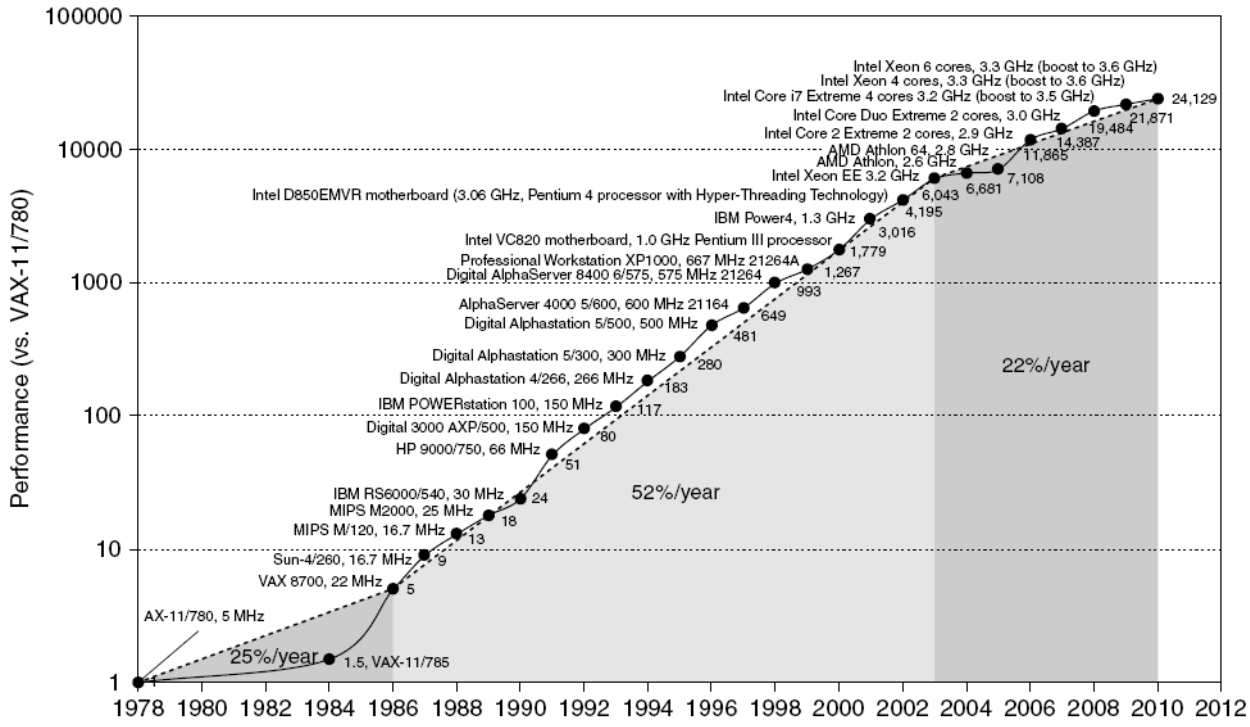
- Harvard Architecture
  - Separate memory holds data and instructions, each are transferred to the CPU through separate buses.



- What is the benefit?
  - Data access corresponding to previous instruction can be performed while fetching current instruction (in the same clock cycle).
  - Widely used in Digital Signal Processors (DSP). Now it's found in most modern processors.

Single Processor Performance

- A minicomputer released in 1977 by Digital Equipment Company (DEC)
- Using transistor-transistor logic (TTL), i.e. bipolar junction transistors (BJTs)
- Operating with a 5MHz clock rate!
- Cost around $150k



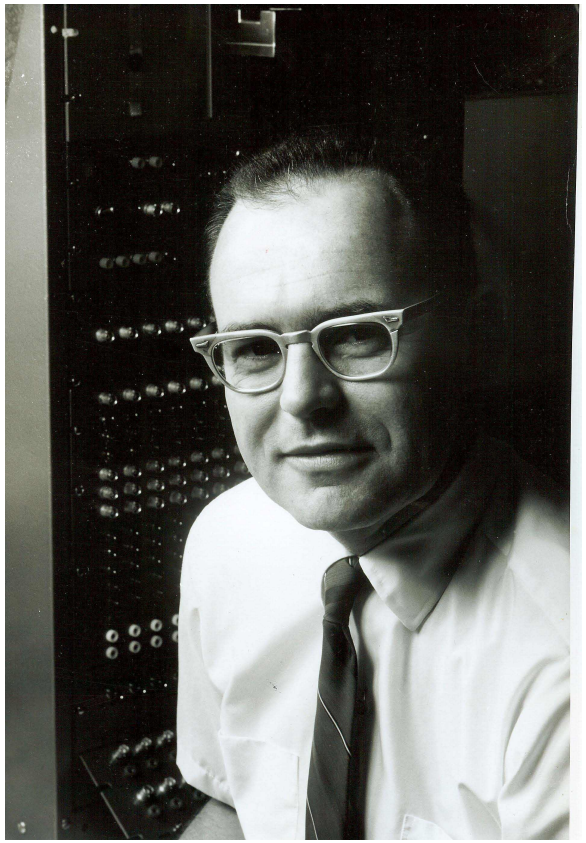
SEMICONDUCTOR TECHNOLOGY

- Switching devices that implement the digital logic have gotten smaller, faster, and more efficient
  - Starting with vacuum tubes
  - Moving on to BJTs and then Metal Oxide Field
  - Settling on to Complementary MOSFETs (CMOS) on Integrated Circuits (ICs)

- Two primary rules of thumb:
  - Moore's Law
  - Dennard's Scaling

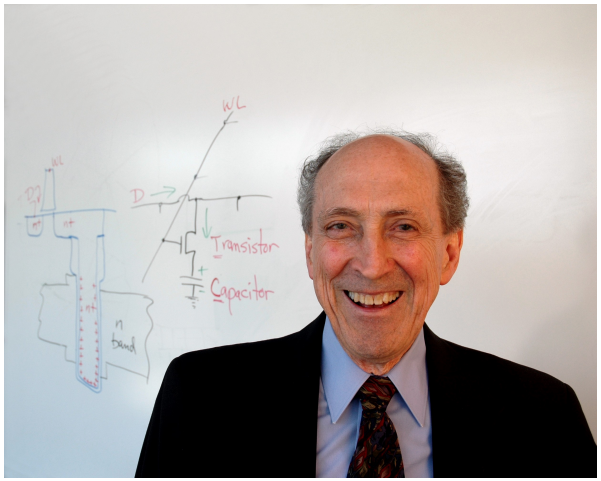
Moore's Law

- In 1965, **Gordon Moore**, the co-founder of Intel, predicted that the number of transistors on an IC would double every year
- In 1975, Moore's law was amended to doubling every *two* years
- Recently, Moore's law has ended!



Moore's Law

- In 1974, Robert Dennard observed that the power density of a given area of silicon was held nearly constant even as the number of transistors increases
- This was due to voltages/currents per transistor going down
- Dennard's scaling ended around 2004 leading to a concept called *Dark Silicon*



Review

key features of computer architecture

Fundamentals of Computer Architecture

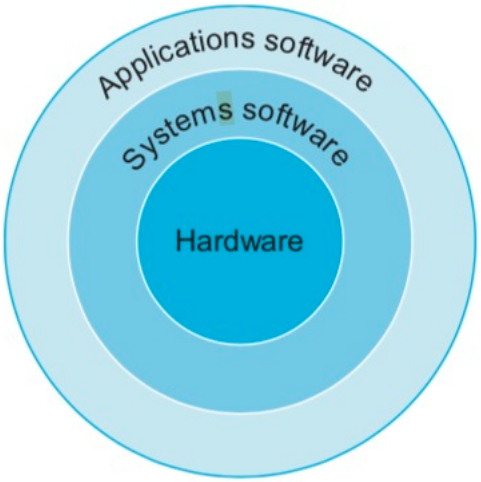
- Moore's Law
  - Prediction made by Gordon Moore in 1965
  - Number of transistors on a chip approximately doubles every 18-24 months
  - Designers must anticipate rapid advancements during a design
- Abstraction after abstraction!
  - Lower-level details hidden from higher levels
  - Necessary to manage complexity
  - Examples: API, ABI, ISA, digital logic
- Make the common fast
  - Optimize the design for the common case
  - Experimentation and measurement identify common cases
  - Often common equals simple
  - Amdahl's Law!

Fundamentals of Computer Architecture

- Performance via parallelism
  - Take advantage of transistors due to Moore's Law
  - Pipelining and superscalar – Instruction-Level Parallelism
  - Multicore – Thread-Level Parallelism
  - Graphics Processing Unit – Data-Level Parallelism
- Memory hierarchy
  - Users want large, fast memory!
  - Large memory (DRAM) is slow but small memory is fast (SRAM)
  - A hierarchy of memory provides the illusion of large, fast memory
- Performance via branch prediction
  - Often times it's faster to predict and recover if incorrect
  - Reduce time waiting on dependencies
  - This is the key to modern processor design

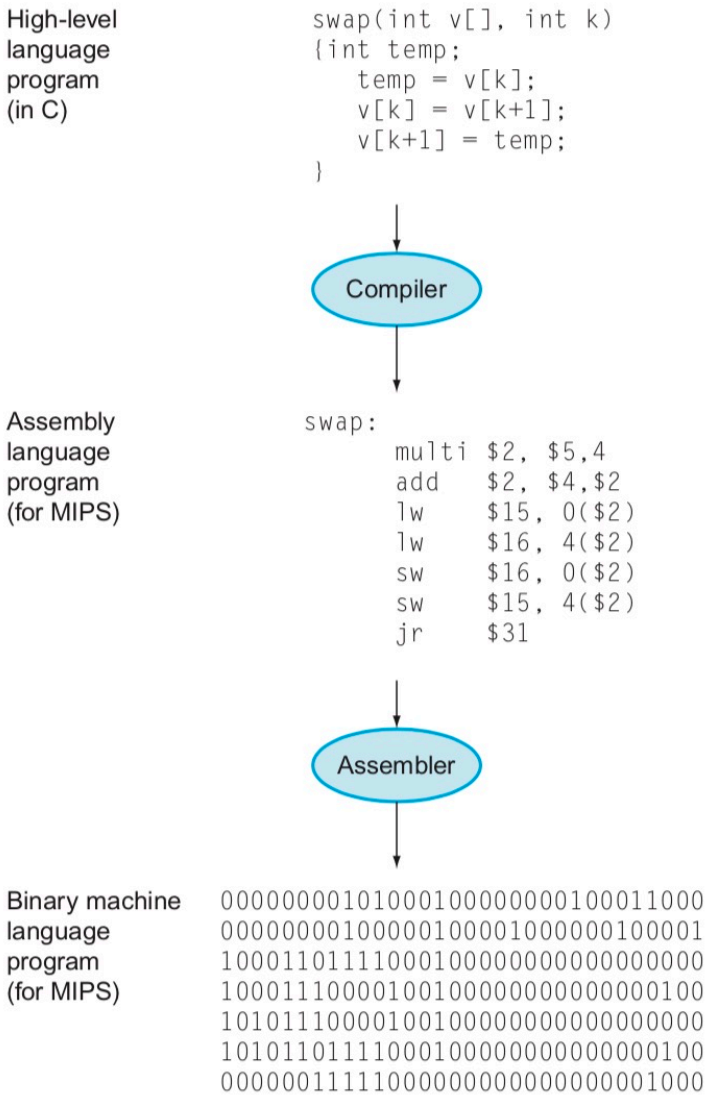
Below Your Program

- Application software
  - Written in High-Level Language (HLL)
  - Implements algorithms and user interface
- Compiler and Libraries
  - Translates HLL code into machine code
  - Provides source for common routines like printf
- Operation System
  - Handling input/output
  - Managing memory and storage
  - Scheduling tasks & sharing resources
- Hardware
  - Processor, Memory, I/O…



Levels of Program Code

- **High-level language**
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
- **Assembly language**
  - Textual representation of instructions
  - Typically one-to-one, except for pseudo instr.
- **Hardware representation**
  - Binary encoded instructions and data
  - Directly loaded into memory and read by machine



Programming Interfaces

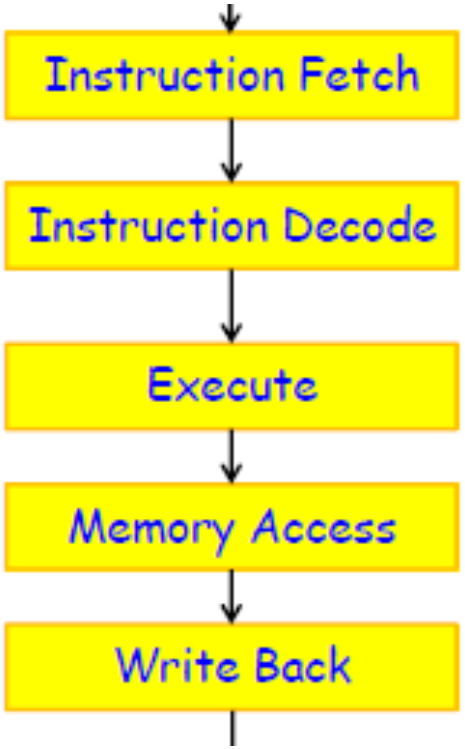
- Application Programming Interface (API)
  - How to interact with different parts of code or libraries at the C level
- Application Binary Interface (ABI)
  - How to interact with the processor and operating system at the assembly level
- Instruction Set Architecture (ISA)
  - The HW/SW boundary!

TOOLS OF THE TRADE

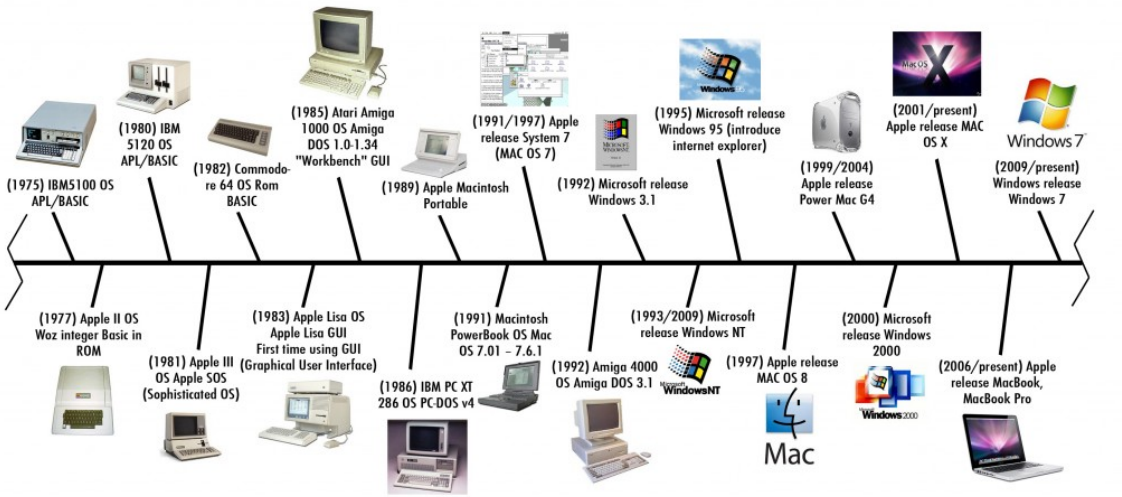
- Hardware Description Language (HDL)
  - Verilog HDL and System Verilog
  - VHDL (Very High Speed Integrated Circuit HDL)
  - SystemC
- Simulation
  - C/C++ architectural simulators such as GEM 5
  - HDL simulators ModelSim
  - Gate-level simulation with SPICE
- Emulation with Field Programmable Gate Arrays (FPGAs)
- Benchmarks!

Program Execution in Computers– Example

- Instruction cycle: processing for execution of an instruction.
  - Instruction fetch (IF): Fetch instruction from memory and get ready to fetch the next Instruction.
  - Instruction decode (ID): Decodes instruction, generate control signals and fetch register operand from register file.
  - Execute (EX): Execute the ALU operation as specified in the opcode of the instruction.
  - Memory access (MA): Perform read/write for load/store operations.
  - Write back (WB): Write the result back to register file.
  - Go to step 1 to execute the next instruction.



The Computer Revolution



Agricultural Age

↓

Industrial Age

↓

Information Age

Computers lead to the third revolution in our civilization

Computing system	Mainframe	Mini computer	Personal computer	Embedded computer
Era	1950s on	1970s on	1980s on	2000s on
Form factor	Multi-cabinet	Multi-board	Single board	Single chip
Resource type	Corporate	Departmental	Family	Personal
Users/system	100s – 1000s	10s – 100s	1s	1/10s
Cost	$ 1 million +	$ 100Ks +	$1Ks – $10Ks	$1s – $100s
Total units	10Ks +	100Ks +	1 billions +	1 Trillions +

Architectural Improvements

Performance is primarily about doing multiple "things"

Parallelism

- Applications exhibit Parallelism
  - Data-Level Parallelism (DLP): Many data items operated on at the same time
  - Task-Level Parallelism (TLP): Separate units of work that can operate independently and at the same time

Architectural Improvements

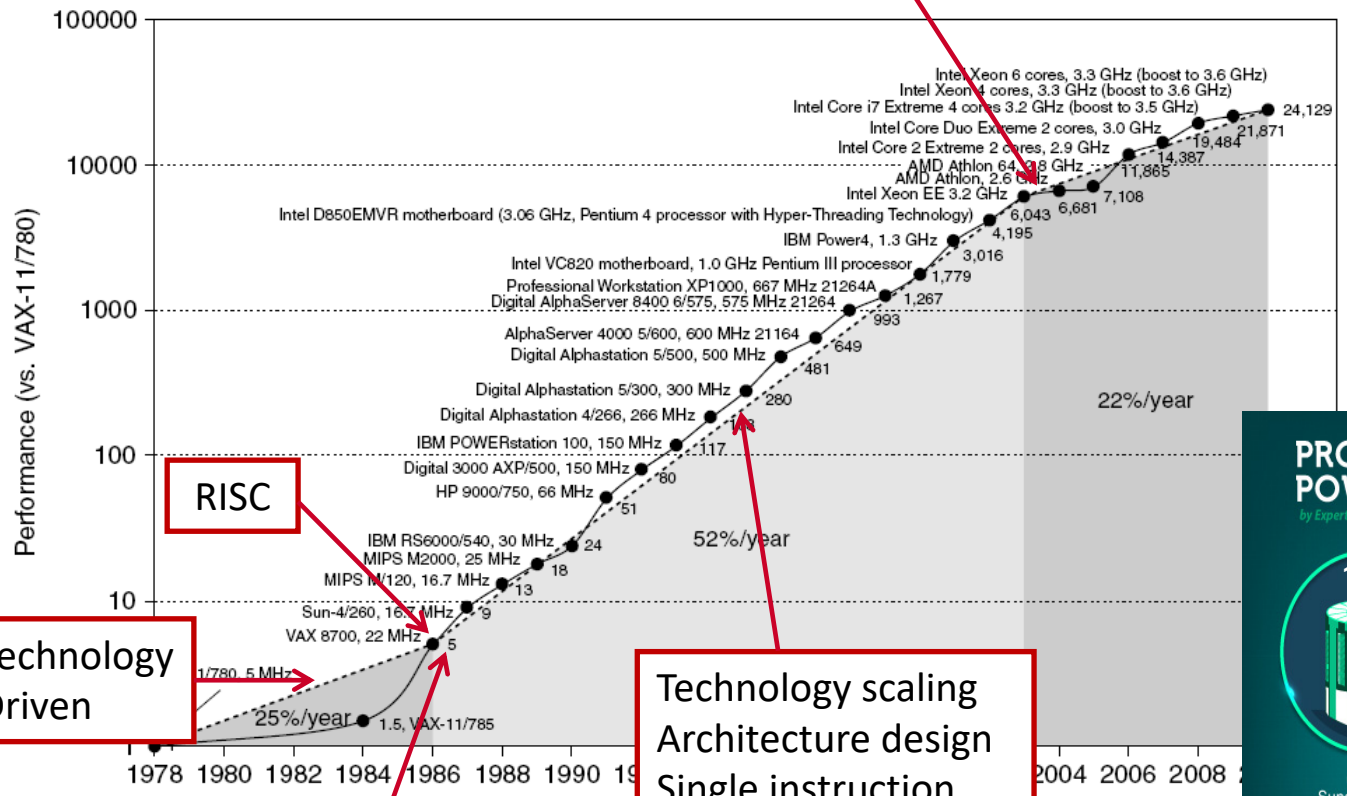
Performance is primarily about doing multiple "things"

Parallelism

- System architectures exploit Parallelism
  - Instruction-Level Parallelism (ILP): Data-level parallelism exploited in the same instruction stream
  - Thread-Level Parallelism (TLP): Data or task-level parallelism exploited by separate instruction streams that may interact
  - Request-Level Parallelism (RLP): Largely independent tasks for separate request streams (users)
  - Vector/GPU architectures: Data-level parallelism exploited by single instruction applied on multiple data items

Growth of Processor Performance

Power limits single stream Technology faces limits Chip multiprocessors (multicore) "Age of parallelism" → Move to multi-processor

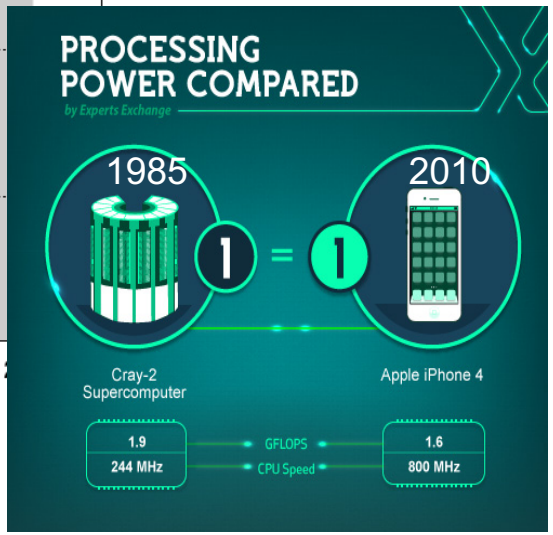


RISC

Technology Driven

Age of the "Killer micros"

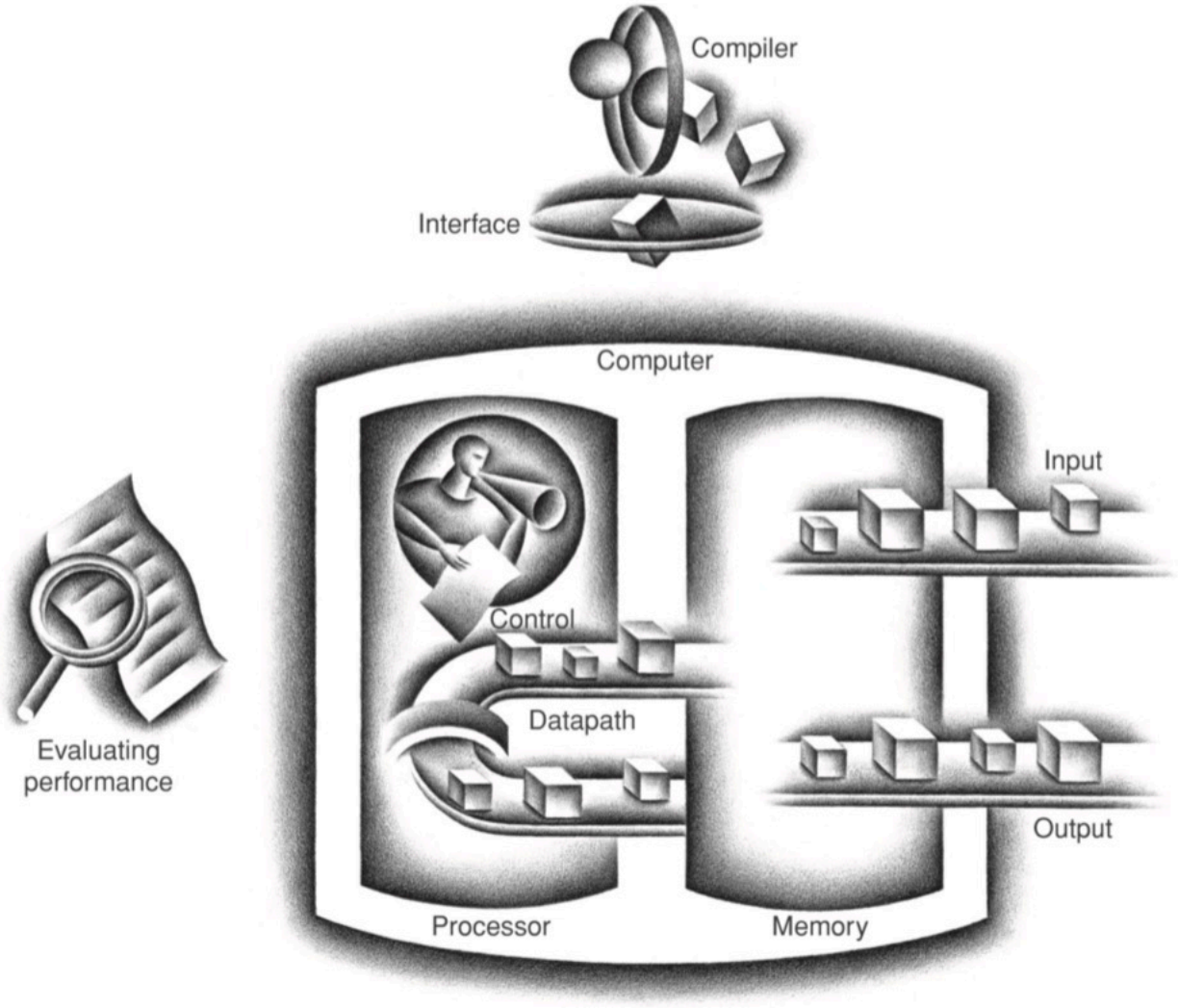
Technology scaling
Architecture design
Single instruction stream "Golden age of performance"



Performance / Cost Improvements

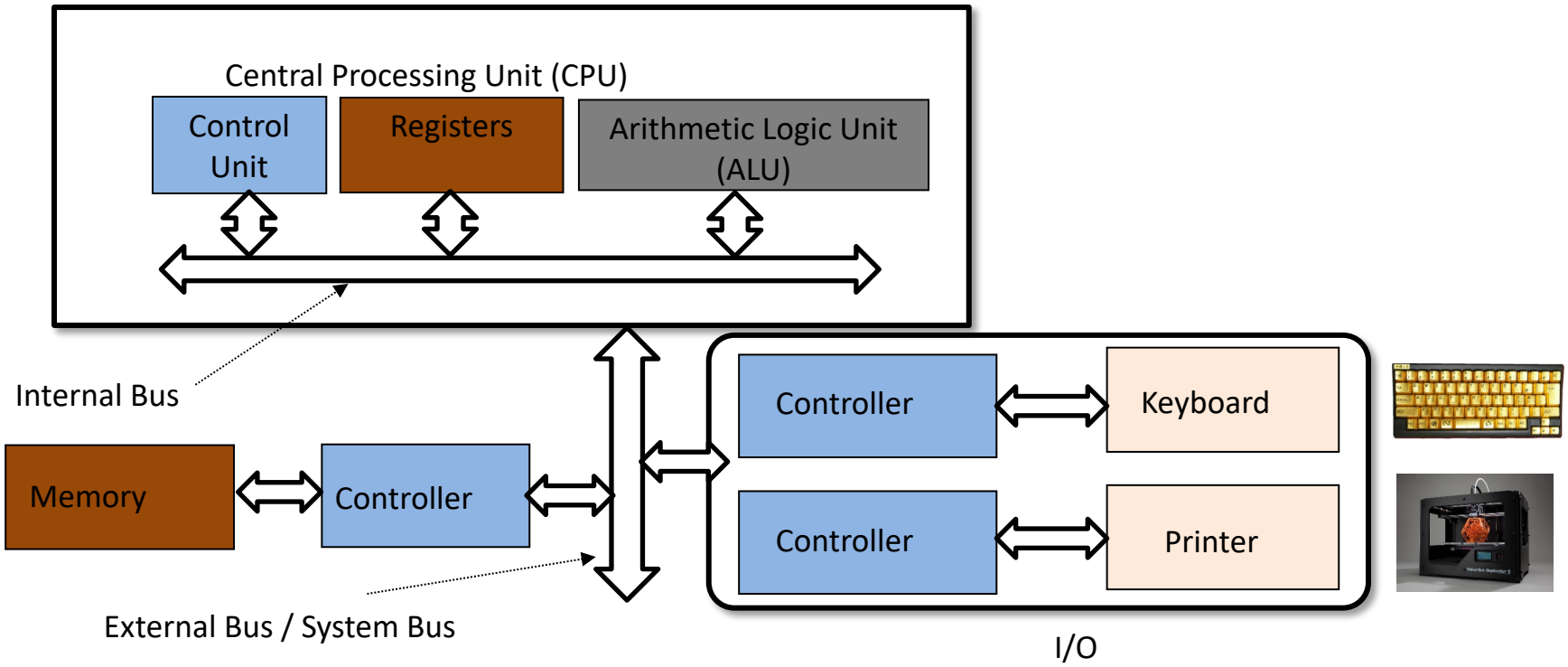
Year	Technology used in computers	Relative performance/unit cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit	900
1995	Very large-scale integrated circuit	2,400,000
2013	Ultra large-scale integrated circuit	250,000,000,000

Components of a Computer



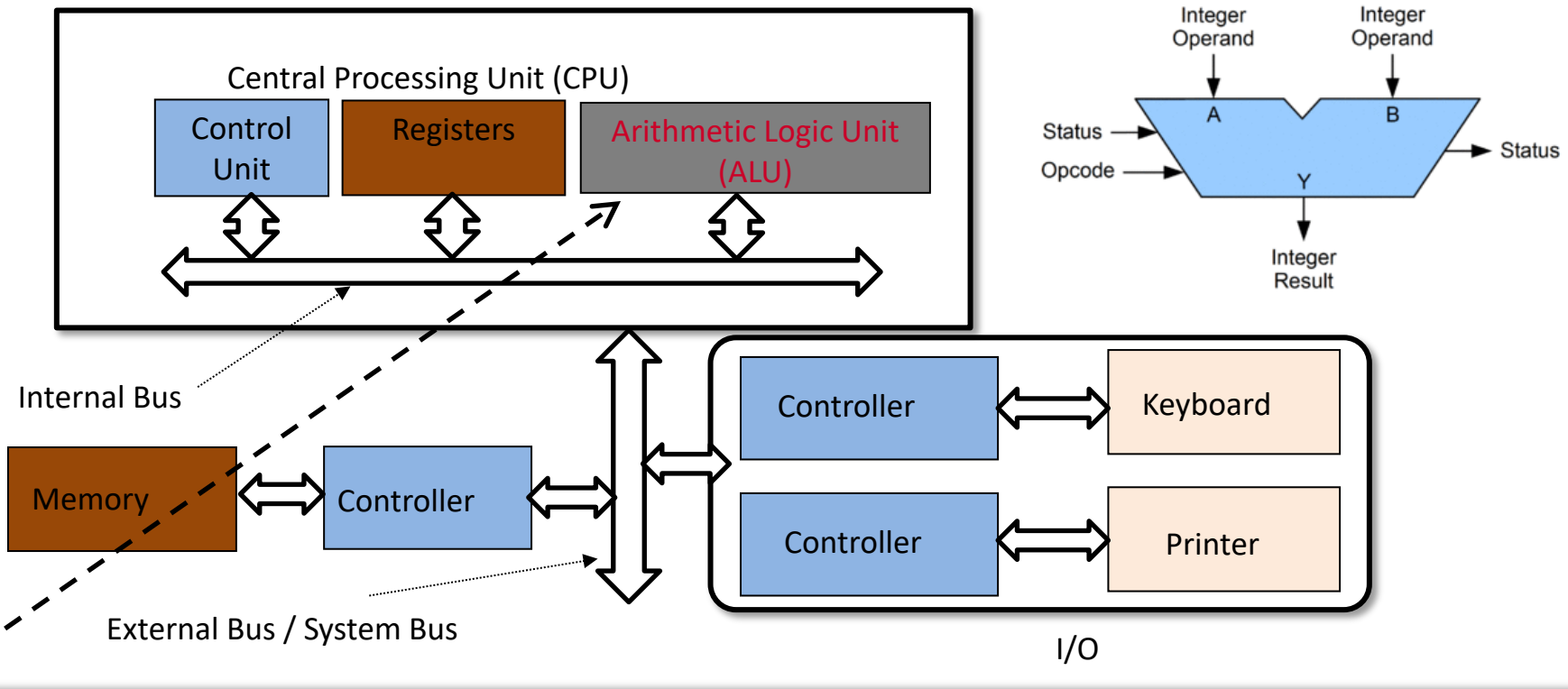
REVIEW KEY FEATURES OF COMPUTER ARCHITECTURE

- 5-basic hardware components of a computer



REVIEW KEY FEATURES OF COMPUTER ARCHITECTURE

- Basic Components: ALU/Computing

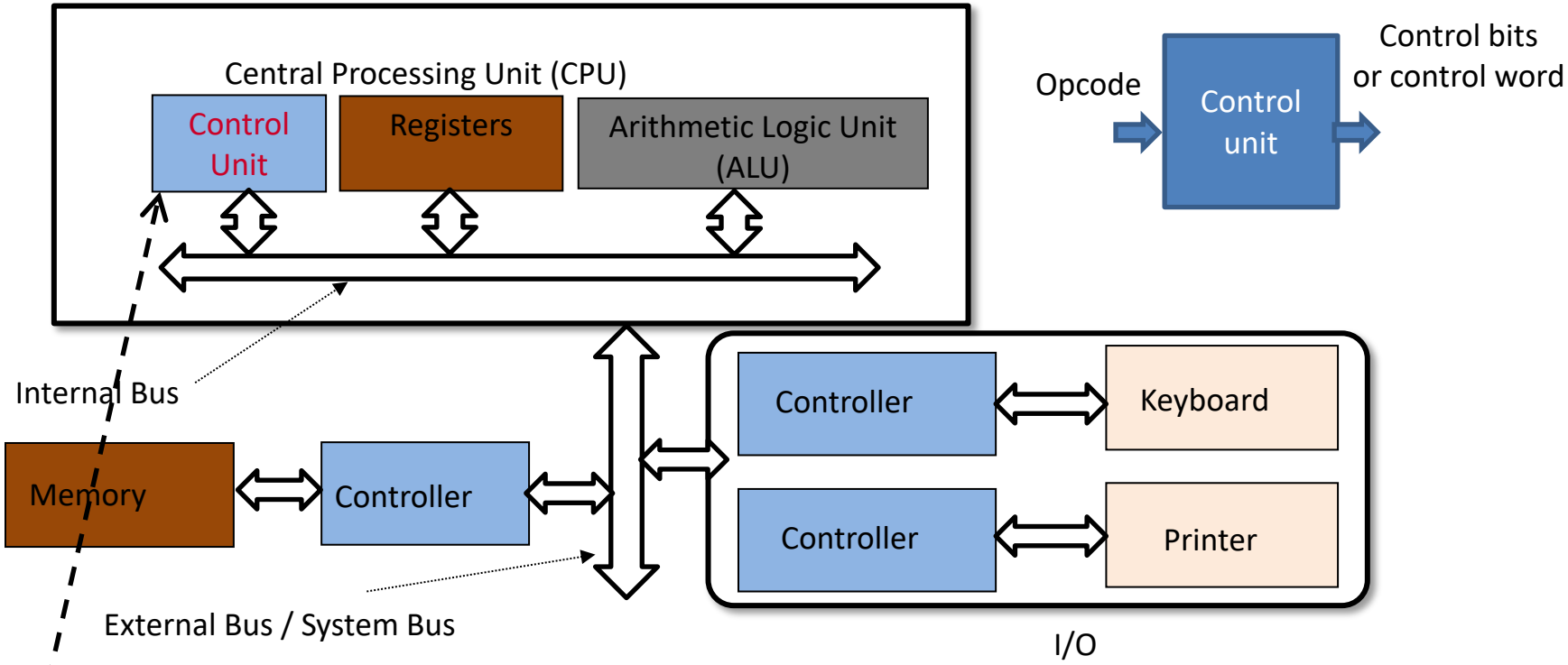


ALU performs integer arithmetic operations and logical operations such as add, subtract, multiply, divide, AND, OR, XOR, NOT etc.

Its input/output is from/to registers, connected directly or through a fast bus - fixed point numbers only.

REVIEW KEY FEATURES OF COMPUTER ARCHITECTURE

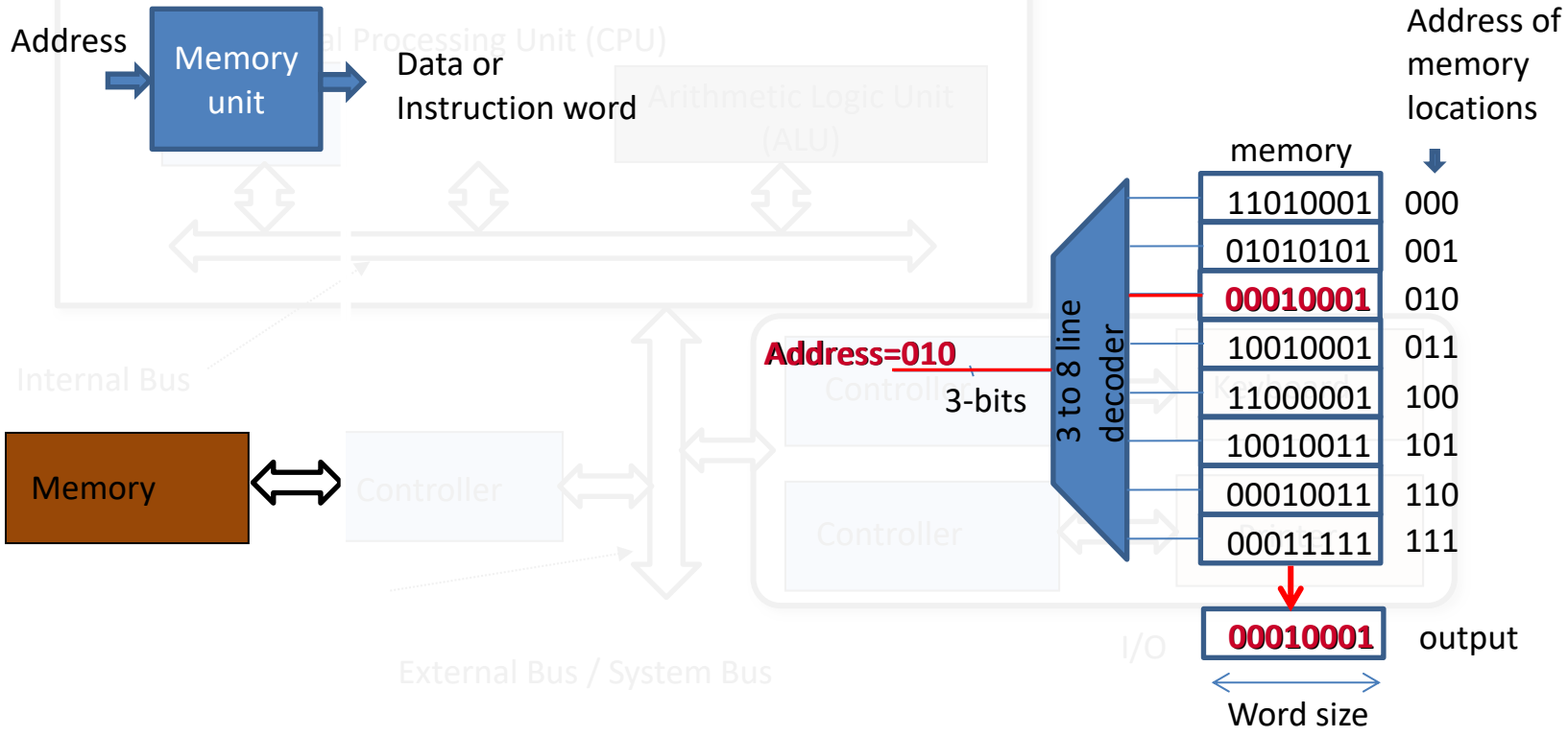
- Basic Components: Control



Control unit generates control signals for data movement and data storage operations, and/or to let the ALU perform the operations specified in the instruction.
The control signals can be generated either using hardware or using microprogram.

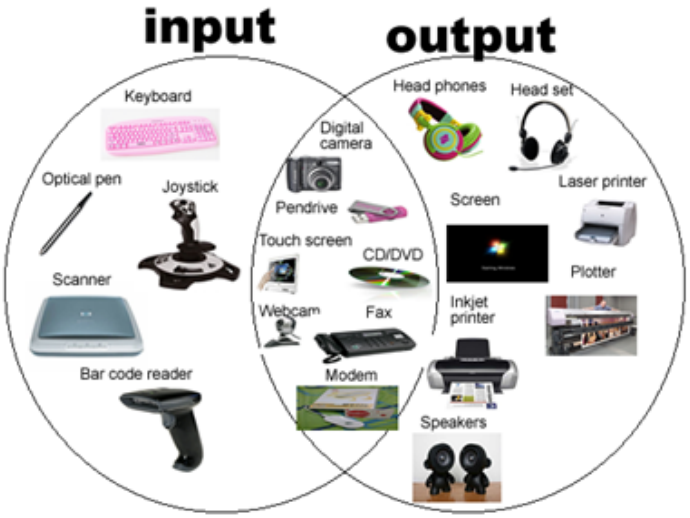
REVIEW KEY FEATURES OF COMPUTER ARCHITECTURE

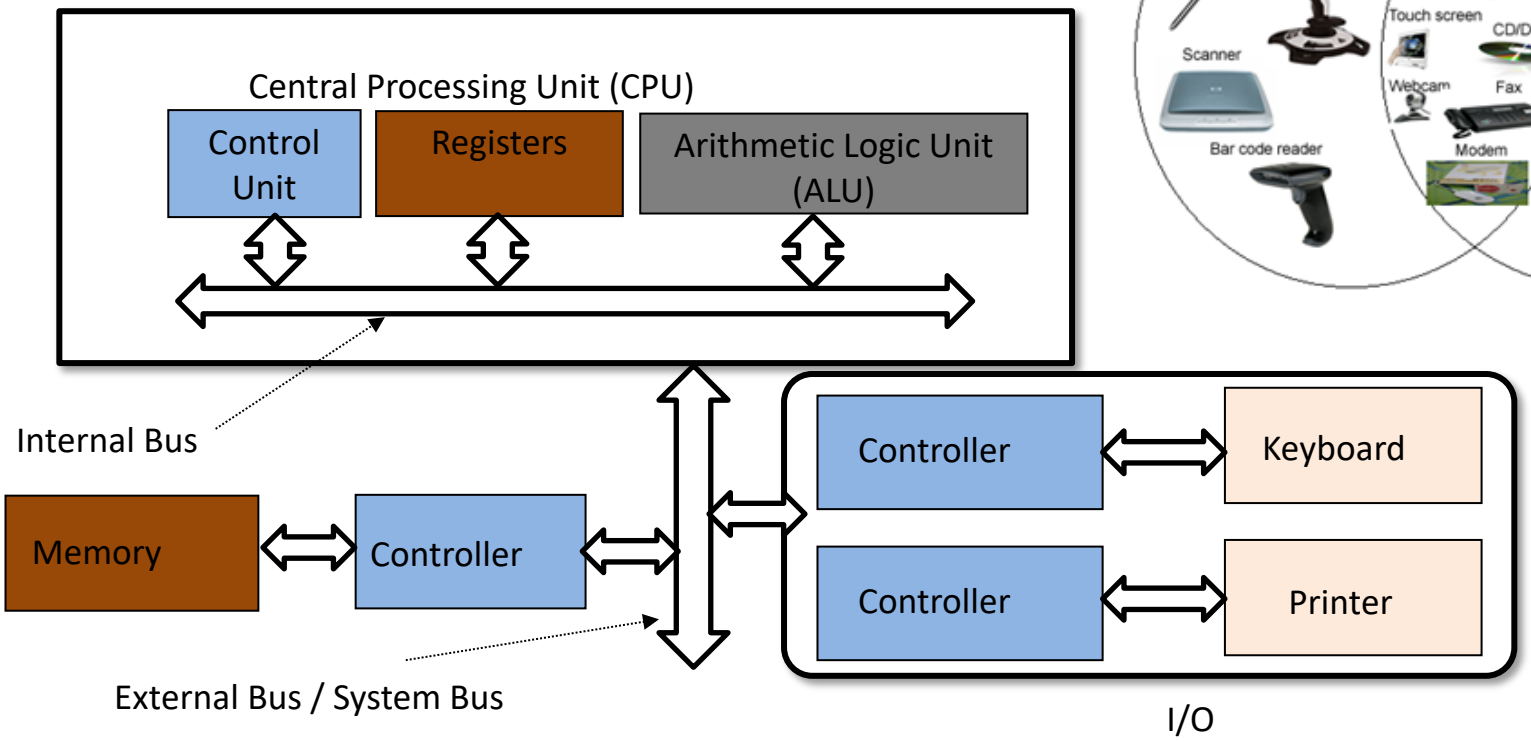
- Basic Components: Memory

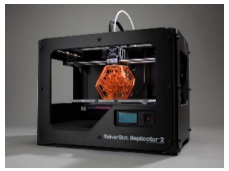


Review key features of computer architecture

- Basic Components: I/O

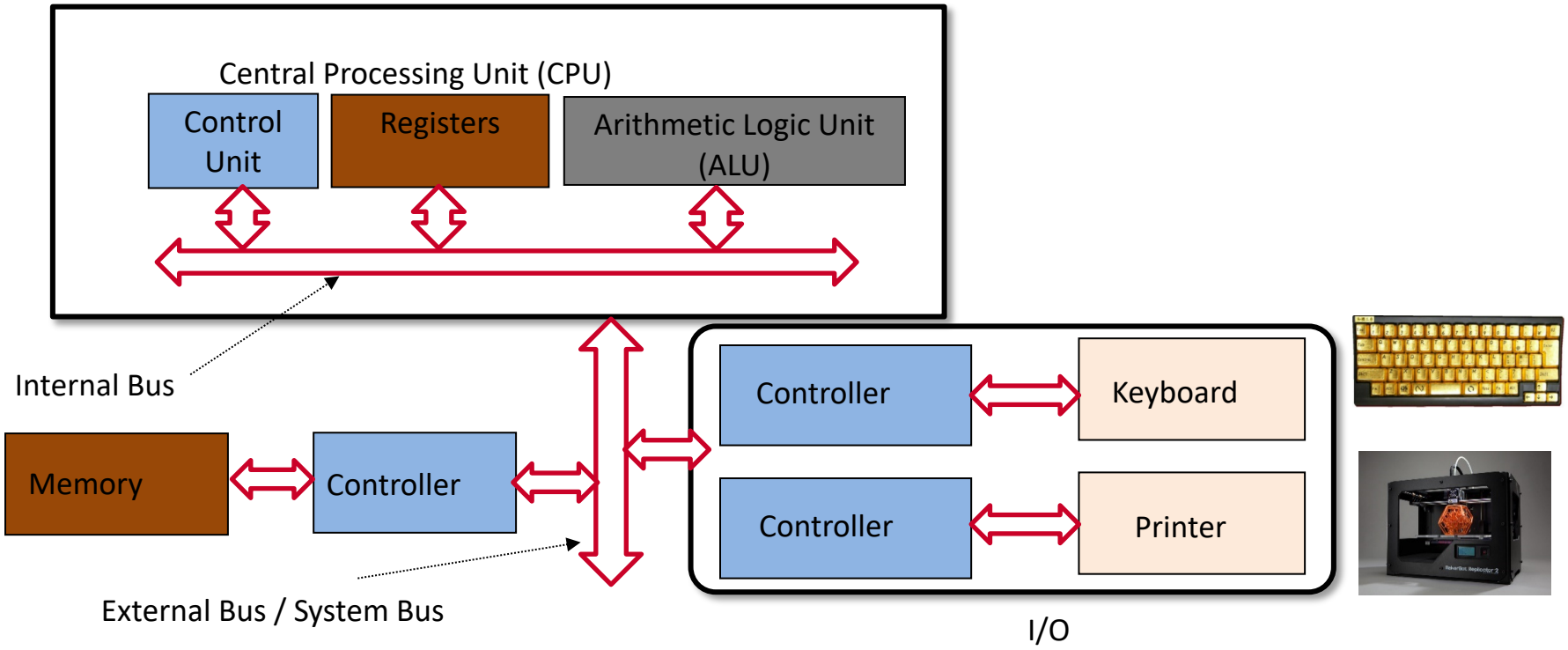






Review key features of computer architecture

- Basic Components: Bus / Communication



What is Computer Architecture?

- Computer architecture deals with design and implementation of computer hardware. There are three main subcategories:
  - *Instruction Set Architecture, or ISA*: ISA defines the machine code that a processor reads and acts upon as well as the word size, memory address modes, processor registers, and data type.
  - *Microarchitecture, or computer organization:* describes implementation of the ISA. It specifies the execution of instruction through the control, storage and computing.
  - *System Design:* specifies all of the hardware components within a computing system, their functionalities and interconnection.

The Instruction Set Architecture (ISA)

- The hardware/software boundary
- Defining the set of instructions a given machine supports
- Two main categories
  - Complex Instruction Set Computer (CISC), e.g. x86, VAX, 68000
  - Reduced Instruction Set Computer (RISC), e.g. MIPS, ARM, RISC-V
- Number of general purpose registers (GPRs)
  - Recent x86 has 16 GPRs
  - Most RISC ISAs have 32 GPRs
- Register-memory vs. Load-store
  - The latter only performs arithmetic-logic operations on data in registers and provides load/store instructions to move data between memory and registers
  - The former has arithmetic-logic instructions that specify data from registers and memory as operands

The Instruction Set Architecture (ISA)

- Memory addressing
  - Most ISAs provide for byte addressing
  - Some ISAs, such as MIPS, enforce memory alignment
  - How many bits of memory address?

- Addressing modes
  - Used to specify operands within an instruction
  - Register, immediate, displacement (base+offset)
  - Absolute, two register, PC-relative, autoincrement/decrement

- Types and sizes of operands
  - floating point vs. integer; 8-, 16-, 32-, 64-bit
  - byte (char), half word (unicode), word (integer), long word or double word
  - single-precision, double-precision, x86's extended double precision (80-bits)

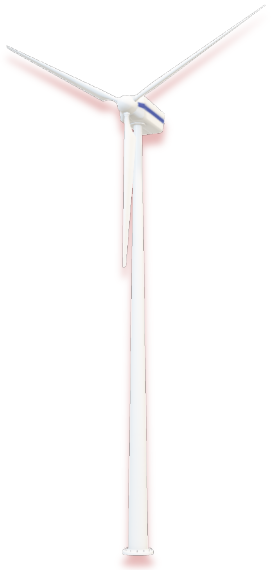
The Instruction Set Architecture (ISA)

- Operations
  - Data transfer such as loads/stores; Arithmetic-logic such as ADD, SUB, AND, XOR ; Floating point; Control flow such as jump, BNE, BEQ
- More on control flow
  - Some ISAs (ARM and most CISCs) rely on conditional codes for branching
  - Branching instructions branch based on state of conditional codes
  - Conditional codes are set during execution of arithmetic-logic operations
- Finally, instruction encoding
  - CISC instructions are typically variable length, e.g. x86 can be 1-15 bytes
  - RISC instructions are typically fixed length at 32bits
- Instruction encoding continued...
  - ARM, MIPS, RISC-V all offer 16-bit extensions to ISA to allow for denser code
  - CISC code size is typically smaller; RISC are easier to decode and pipeline

The Instruction Set Architecture (ISA)

A few comments about ISAs

- Most programmers don't care and shouldn't
- Some deeply embedded applications or high workload situations do care, e.g. multiply-accumulate instructions are good for public-key crypto and vector instructions are good for multimedia
- Computer architect does care! e.g. RISC is easier to decode and pipeline
- x86 literally decodes instructions on fly into RISC-like micro-ops



What did we Learn So far

Thoughts for the day!

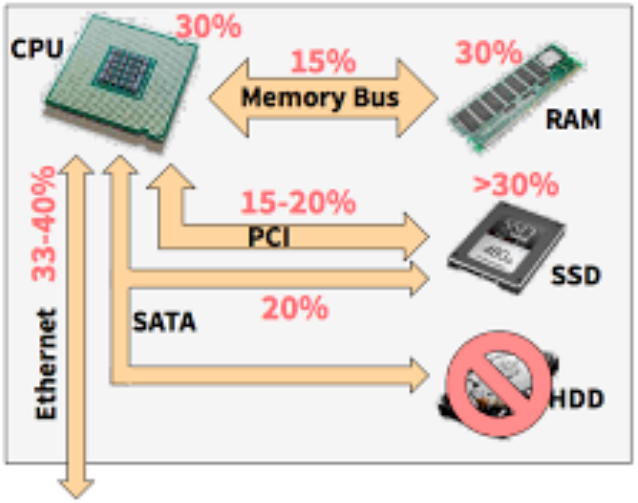
- Programmers' view
  - ✓ ISA
  - ✓ Program Execution
- Architects' view
  - ✓ Storage, Control, Computing, Communication, Peripheral

David Patterson and Carlo Séquin, 1981, inventors of RISC

Performance metrics and performance enhancement techniques

Referred to Chapter 1

TECHNOLOGY TRENDS

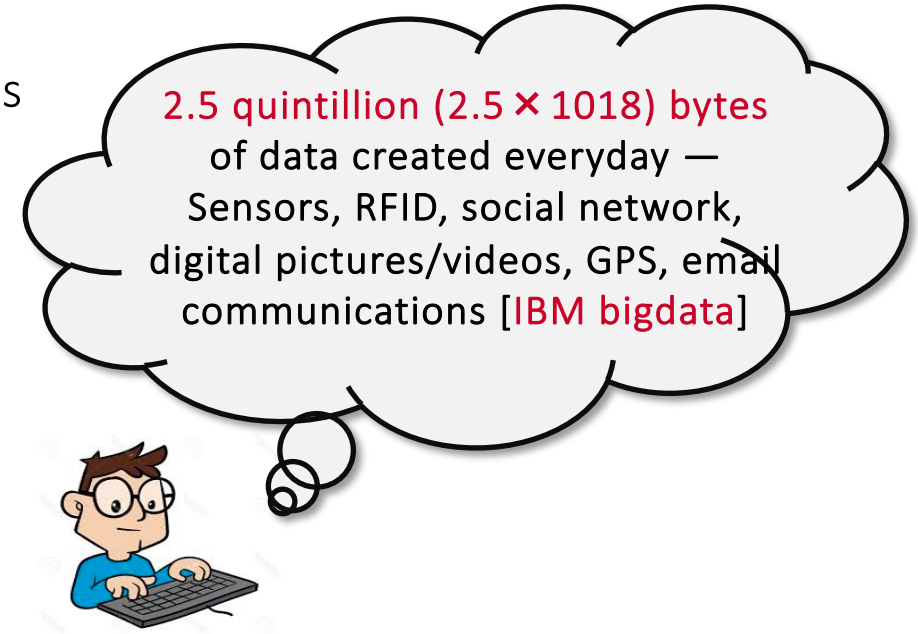


- Integrated Circuit (IC) technology: Moore's Law
  - Transistor density: 35%/year;
  - Die size: 10-20%/year
- DRAM capacity: 25-55%/year (slowing down)
- Flash (EEPROM) capacity : 50-60%/year
  - 15-20x cheaper/bit than DRAM
  - Replacing hard disks in desktops/laptops; but hard disk is 8-10x cheaper
- Magnetic disk capacity: 40%/year
  - 15-25x cheaper/bit than flash; 200-300X cheaper/bit than DRAM
- Bandwidth or throughput
  - 32,000-40,000x improvement for processors; 300-1,200x memory and disks
- Latency or response time:
  - 50-90x improvement for processors; 6-8x for memory and disks

NEED FOR HIGH COMPUTING PERFORMANCE

- Support for better quality of service
  - Support increasing transmission speed
  - Multimedia applications
    - 300 hours of video are uploaded to YouTube every minute!
  - Increasing security need

- Computation-intensive applications
  - DNA sequence analysis
  - Scientific computing
  - Weather forecasting
  - Big data analytics



AVAILABILITY OF COMPUTING RESOURCES: MOORE'S LAW

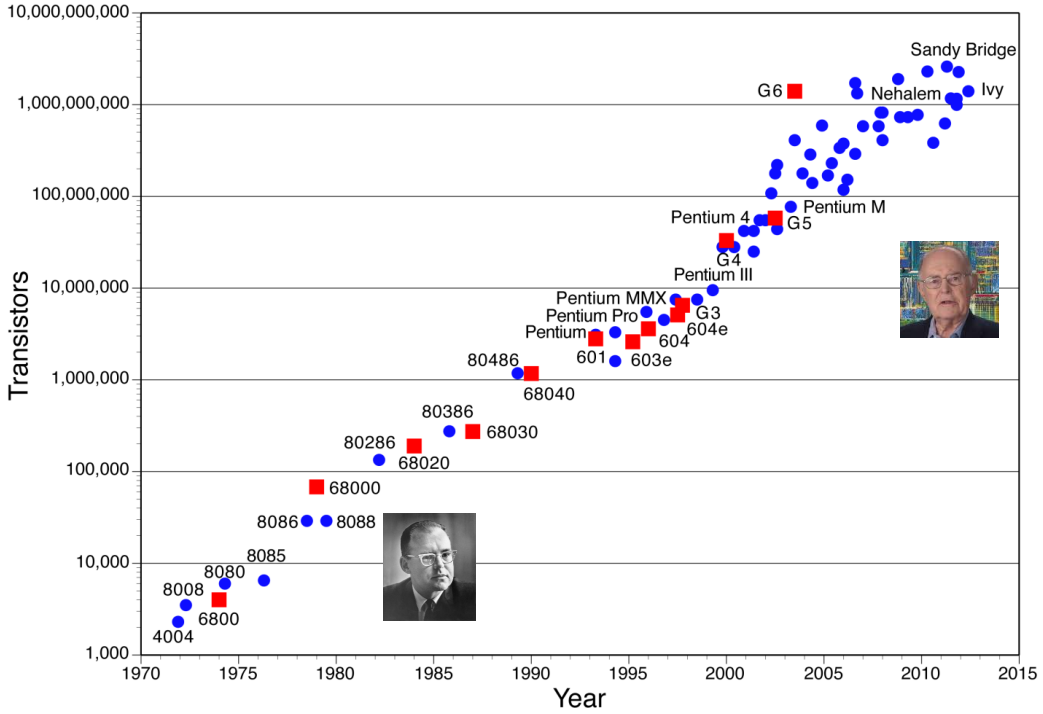
- Smaller transistors and shorter wires are faster! Why?

- Computer architectures have more logic to use:
  - ✓ 4-bit, 8-bit, 16-bit, 32-bit, 64-bit microprocessors
  - ✓ More and more on-chip cache
  - ✓ Aggressive pipelining and out-of-order execution
  - ✓ SIMD instructions, creating processors with many Arithmetic Logic Units (ALUs)
  - ✓ Multicore

AVAILABILITY OF COMPUTING RESOURCES: MOORE'S LAW

- **Moore's law (1965):** number of transistors in an IC chip doubles approximately every two years. Named after Gordon E. Moore, co-founder of Intel Corporation.

End of Moore's law??



- ✓ We got double hardware resources every 2 years for 50 years!!
- ✓ What are we going to do with that??
- ✓ Computer architectures to be designed to use them judiciously.

AVAILABILITY OF COMPUTING RESOURCES

- Intel Microprocessor Scaling

Microprocessor	16-Bit address/ bus, microcoded	32-Bit address/ bus, microcoded	5-Stage pipeline, on-chip I & D caches, FPU	2-Way superscalar, 64-bit bus	Out-of-order 3-way superscalar	Out-of-order superpipelined, on-chip L2 cache	Multicore OOO 4-way on chip L3 cache, Turbo
Product	Intel 80286	Intel 80386	Intel 80486	Intel Pentium	Intel Pentium Pro	Intel Pentium 4	Intel Core i7
Year	1982	1985	1989	1993	1997	2001	2015
Die size (mm^2)	47	43	81	90	308	217	122
Transistors	134,000	275,000	1,200,000	3,100,000	5,500,000	42,000,000	1,750,000,000
Processors/chip	1	1	1	1	1	1	4
Pins	68	132	168	273	387	423	1400
Latency (clocks)	6	5	5	5	10	22	14
Bus width (bits)	16	32	32	64	64	64	196
Clock rate (MHz)	12.5	16	25	66	200	1500	4000
Bandwidth (MIPS)	2	6	25	132	600	4500	64,000
Latency (ns)	320	313	200	76	50	15	4

AVAILABILITY OF COMPUTING RESOURCES

- DRAM and Network Scaling

Memory module	DRAM	Page mode DRAM	Fast page mode DRAM	Fast page mode DRAM	Synchronous DRAM	Double data rate SDRAM	DDR4 SDRAM
Module width (bits)	16	16	32	64	64	64	64
Year	1980	1983	1986	1993	1997	2000	2016
Mbits/DRAM chip	0.06	0.25	1	16	64	256	4096
Die size (mm^2)	35	45	70	130	170	204	50
Pins/DRAM chip	16	16	18	20	54	66	134
Bandwidth (MBytes/s)	13	40	160	267	640	1600	27,000
Latency (ns)	225	170	125	75	62	52	30
Local area network	Ethernet	Fast Ethernet	Gigabit Ethernet	10 Gigabit Ethernet	100 Gigabit Ethernet	400 Gigabit Ethernet	
IEEE standard	802.3	803.3u	802.3ab	802.3ac	802.3ba	802.3bs	
Year	1978	1995	1999	2003	2010	2017	
Bandwidth (Mbits/seconds)	10	100	1000	10,000	100,000	400,000	
Latency (µs)	3000	500	340	190	100	60	

AVAILABILITY OF COMPUTING RESOURCES

- Hard Disk Scaling

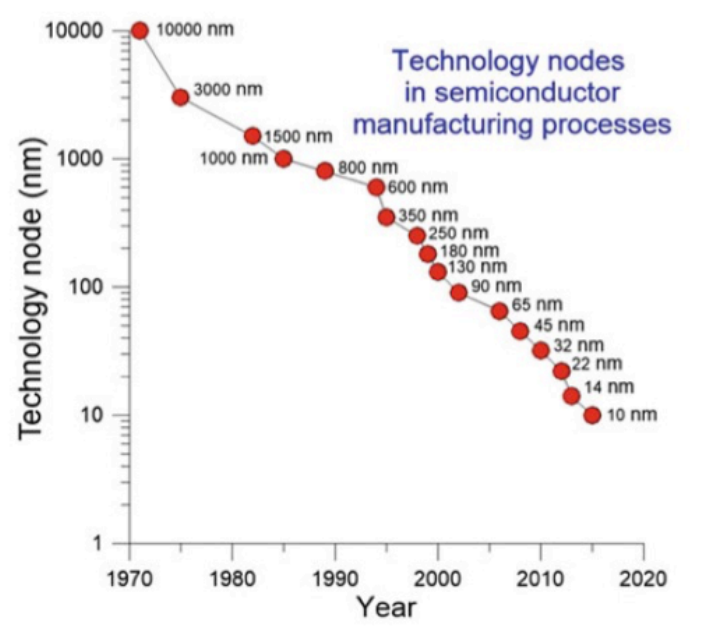
Hard disk	3600 RPM	5400 RPM	7200 RPM	10,000 RPM	15,000 RPM	15,000 RPM
Product	CDC WrenI 94145-36	Seagate ST41600	Seagate ST15150	Seagate ST39102	Seagate ST373453	Seagate ST600MX0062
Year	1983	1990	1994	1998	2003	2016
Capacity (GB)	0.03	1.4	4.3	9.1	73.4	600
Disk form factor	5.25 in.	5.25 in.	3.5 in.	3.5 in.	3.5 in.	3.5 in.
Media diameter	5.25 in.	5.25 in.	3.5 in.	3.0 in.	2.5 in.	2.5 in.
Interface	ST-412	SCSI	SCSI	SCSI	SCSI	SAS
Bandwidth (MBytes/s)	0.6	4	9	24	86	250
Latency (ms)	48.3	17.1	12.7	8.8	5.7	3.6

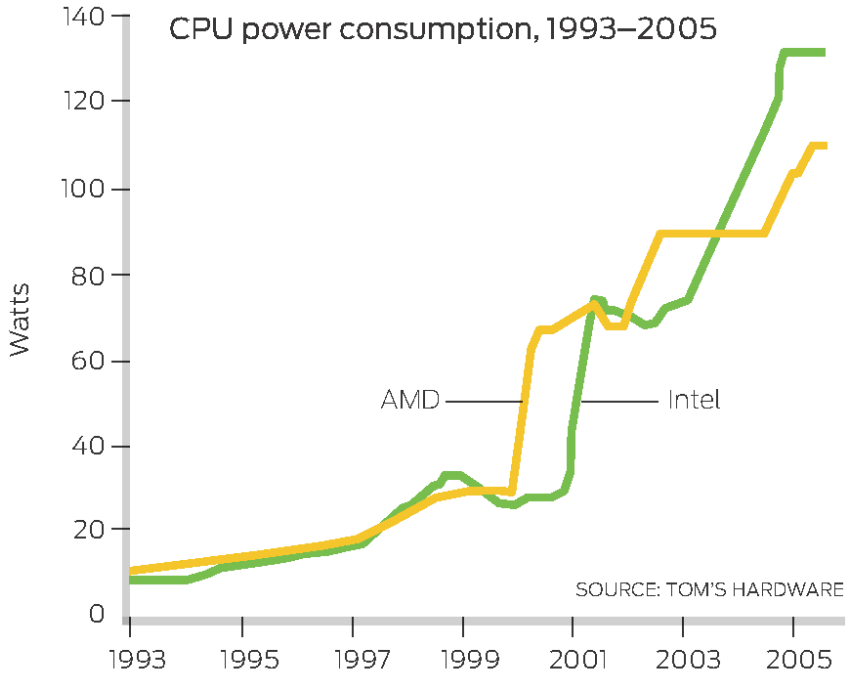
PERFORMANCE POWER AND TECHNOLOGY SCALING

- Transistors are getting smaller and faster with technology scaling

➔ More logic can be integrated to a small area and thus high performance

- Higher performance accompanied by more power consumption.





Reference: P.E. Ross, Why CPU Frequency Stalled, IEEE Spectrum, Vol 45, no 4, 2008, Page(s): 72

Design Goals and Constraints

- Functional requirement: must process data according to the instructions.
  - tests and verifications are required at different stages, since it is not possible to modify the processor once fabricated.
- Reliability: should continue to perform correctly.
  - important for all modern computers and mission critical applications.
  - reliable and fault tolerant computing is an important area of study, which considers various approaches to improve fault tolerance and reliability.
- Cost: is a very important factor.
  - embedded consumer products are particularly highly sensitive to cost.
- Performance: is a basic requirement.
  - a computing system need to provide the desired performance.
- Power consumption: is a very important requirement nowadays

Performance VS **power consumption**

Evolution of computer architecture is driven to improve these two goals.

PERFORMANCE: METRICS & ENHANCEMENT TECHS

- What is performance?
  - Performance indicator - ***Execution time:*** is the time to execute a program
    - Minimize elapsed time for program=$time_{end} - time_{start}$
    - Called response time (execution time)
    - Less the execution time-> better is the performance
    - $Performance = \frac{1}{Execution\ Time}$

- Key features of high-performance architectures
  - Memory hierarchy: multi-level cache design for fast, cost-effective memory
  - Pipeline and parallel instruction execution, super-scale processors
  - Data parallel computing: GPU, vector, and SIMD machines
  - Multicore processors, heterogeneous multi-processor systems on chip
  - Custom computing architectures, and domain-specific architectures

Performance: Metrics & Enhancement techs

- Factors affect execution time?
  - $\boldsymbol{Execution\ Time = IC \times CPI \times T}$
  - Instruction count (*IC*): Application/program; ISA
  - Clock per instruction (*CPI*): ISA; Datapath design; Parallel and pipeline HW design
  - Clock period (*T*): Semiconductor technology; Datapath design and implementation

- Decrease in one may lead to increase in other two.

PERFORMANCE: METRICS & ENHANCEMENT TECHS

- Challenges on performance enhancement
  - ❖ Reduction of clock cycle time (T) / increase clock frequency
  - Power consumption increases with increase in clock frequency
  - Memory operations take longer than a clock period->***memory-wall problem***(memory is relatively slower than CPU, CPU wait for data/instructions)

  - ❖ Reduction of instruction count (IC)
  - More complex instructions
  - Multi-issue processor: VLIW/superscalar, SIMD, vector processor

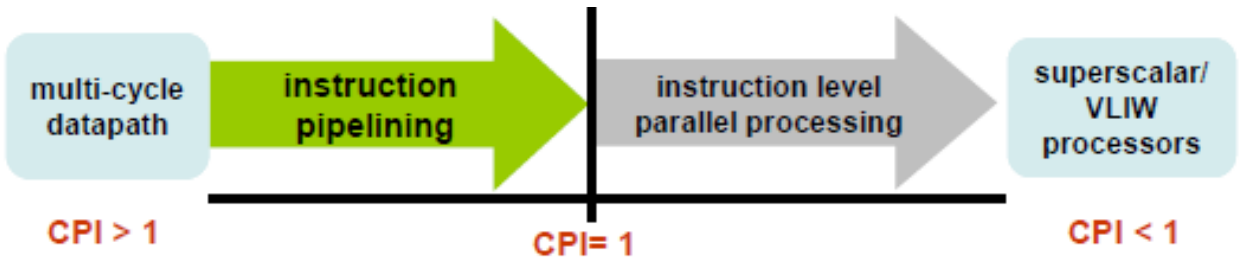
  - ❖ Reduction of cycle per instruction (CPI)
  - Instruction pipelining: Pipeline datapath
  - Multi-issue processor: VLIW/superscalar processor

PERFORMANCE: METRICS & ENHANCEMENT TECHS

- Pipeline

Instr. No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7

- CPI for different datapath and processors
  - Multi-cycle datapath
  - Pipeline datapath
  - Parallel datapath
  - Superscalar/VLIM processors



Performance Speedup

- Case 1: speedup of computer-A over computer-B can be computed as

$$Speedup = \frac{Perf_A}{Perf_B}, Speedup = \frac{Time_B}{Time_A}$$

- Case 2: speedup achieved due to enhancement technique can be as

$$Speedup = \frac{T_{unenhanced}}{T_{enhanced}} = \frac{T_{original}}{T_{enhanced}}$$

Performance Speedup

- Example: If fraction E of the program is enhanced by a factor of S in an enhanced machine, then determine the speedup of enhanced machine over the machine before enhancement (original machine). What is the maximum speed up for a given E if S can be any value greater than or equal to 1?

Fraction $U = (1 - E)$ of the program is not enhanced.

if T is execution time of program in unenhanced (original) machine:

- Time required for the execution of unenhanced fraction $= T \times (1 - E)$
- Time required for the execution of enhanced fraction $= [T \times E]/S$
- Total execution time in the enhanced machine

$$T' = [T \times (1 - E)] + [T \times E]/S$$

$T' = T \times \left[(1 - E) + \frac{E}{S}\right] = T \times [1 - E(S - 1)/S]$ (check: if $S = 1$ then $T' = T$).

Speed up $= \frac{T}{T'} = \frac{T}{T} \times \left[(1 - E) + \frac{E}{S}\right] = \frac{1}{[(1-E)+E/S]}$

Performance Speedup - Amdahl's Law

- Amdahl's law: the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.

  - If fraction (1 – E) of application cannot be enhanced for parallel implementation, then speedup is limited by a factor of 1/(1-E), even if the rest of the application is infinitely sped up, and involve infinitesimal time for the computation.

  - Amdahl's law defines the ***speedup*** that can be gained by using a particular feature. What is the ***speedup***?

$$Speedup = \frac{Perf.of\ entire\ task\ using\ enhancement\ when\ possible}{Perf.of\ rntire\ task\ without\ using\ enhancement}$$

Performance Speedup - Amdahl's Law

- Amdahl's law gives us a quick way to find the speedup from some enhancement, which depends on two factors:
  - The fraction of the computation time in the original computer that can be converted to take advantage of the enhancement.
  - The improvement gained by the enhanced execution mode, that is, how much faster the task would run if the enhanced mode were used for the entire program.

E is fraction of program , enhanced by a factor S.

$$Execution\ tim_{enhanced} = Execution\ tim_{unenhanced} \times (1 - E + \frac{E}{S})$$

$$Thus, \qquad Speedup_{overall} = \frac{Execution\ tim_{unenhanced}}{Execution\ tim_{enhanced}} = \frac{1}{1 - E + \frac{E}{S}}$$

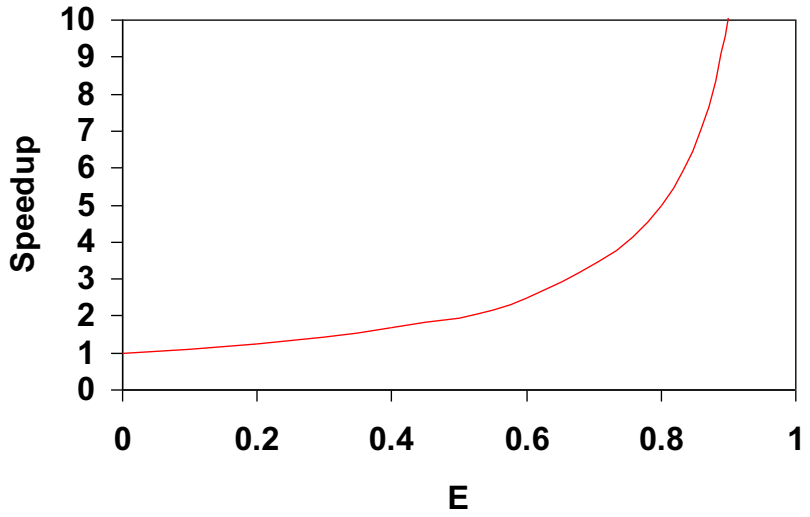
Performance Speedup - Amdahl's Law

- Amdahl's law gives us a quick way to find the speedup from some enhancement, which depends on two factors:

E is fraction of program , enhanced by a factor S. Make common case fast:

$$\lim_{s \to \infty} \frac{1}{1 - E + \frac{E}{S}} = \frac{1}{1 - E}$$

$$Speedup_{max} = \frac{1}{1 - E}$$



EMBARRASSINGLY PARALLEL PROBLEMS

- Amdahl's Law - Fixed Problem Size

Sequential	Parallel

Sequential	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9

$$Speedup_{parallel}(E,S) = \frac{1}{(1-E)+\frac{E}{S}} = \frac{S}{S(1-E)+E} = S$$

What if $E \rightarrow 1$

Power and energy

- There are NOT the same metric
  - Energy is measured in Joules (J)
  - Power is J/s
- Energy per task is better metric for efficiency
  - Relate to battery life in personal mobility device(PMDs)
  - Reduce energy bills in WSC!!!!
  - If processor A consumes 2x the power as processor B but complete the same task in the fourth of the time, there is a 2x gain in energy efficiency!

POWER AND ENERGY

Power: We do care about

- Power limitations
  - Must get power **into** the IC and distribute it
  - And **Out** in the form of heat, e.g, a $2.25cm^2$ die could consume 100W

- Thermal Design Power (TDP)
  - Characterizes sustained power consumption
  - Used as target for power supply and cooling system
  - Lower than peak power (1.5x is typical)
  - Must be higher than average!

POWER AND ENERGY

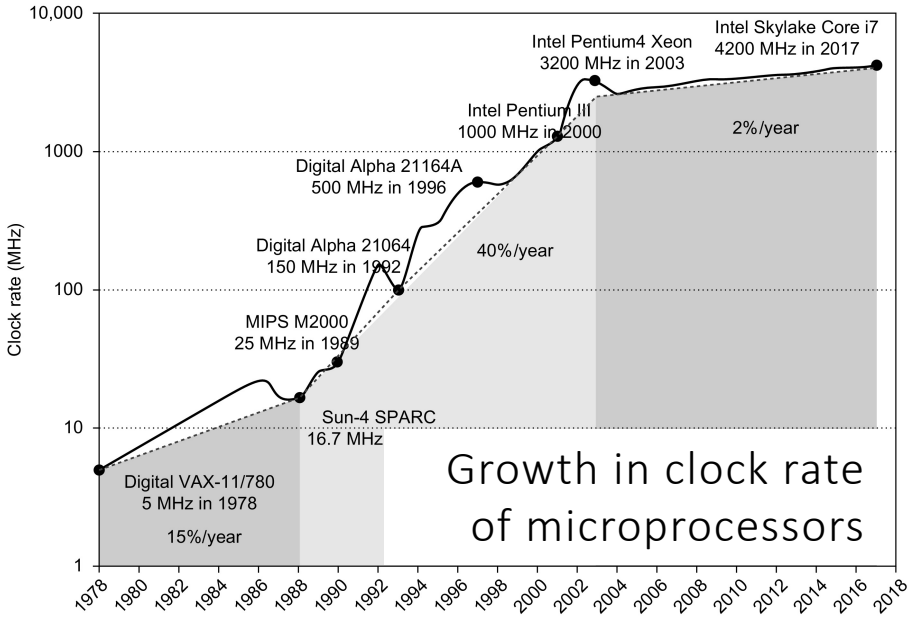
Energy Consumption in CMOS

- Dynamic vs. Static
  - Dynamic: switching of transistors and clock
  - Static: current leakage through imperfect transistors
- Dynamic energy
  - Consider transistor switching from 0->1 or 1->0
  - A transistor gate looks like a capacitor
  - Energy in a capacitor is $\frac{1}{2} \times C \times V^2$
- Dynamic power
  - $P_{dynamic} = \frac{1}{2} \times C \times V^2 \times \alpha \times f$ (α is activity factor to account switch/clock cycle)
- Static power
  - $P_{static} = V \times I_{leakage}$ ($I_{leakage}$ depends on # of transistors and their leakage)

Power Observations

- Dynamic power is linearly related to the clock rate and capacitance but quadratically related to voltage
- Static power is linearly related to leakage current and voltage
- Dennard's scaling implied that *C* and *V* of each transistor would scale down as density and *f* go up

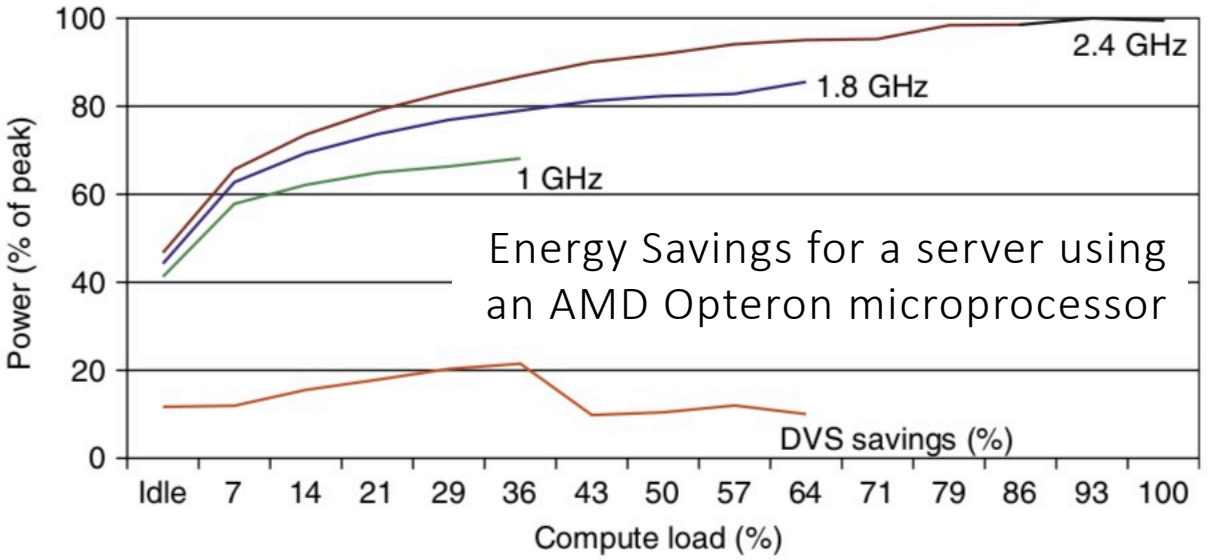
- **Power Examples**
  - Intel 80386 consumed 2W at 16MHz
  - Intel Core i7-6700K consumed 95W at 4.0GHz
  - i7 die is 1.5cm x 1.5cm (2.25cm2)
  - Reaching the limits of what can be cooled by air



Reducing Power and energy

- Lower *f* reduces power but not energy per task! Why?
- Significant power savings can be achieved by reducing *V*. How do we do that?
  - Before Dennard's scaling, moving to a new tech node would get us a lower voltage
  - Now days, slowing f down allows V to be lower. Why?
  - This technique is called **Dynamic Voltage Frequency Scaling** (DVFS)

DVFS on server with AMD Opteron



Reducing Power and energy

- Clock gating
  - Turn off the clock of the portions of unused logic
  - The clock network consumes a significant amount of power
  - Eliminates dynamic power for unused logic
  - Static power is still an issue
- Power gating
  - Turn off power of portions of unused logic
  - Eliminates both static and dynamic power
  - Dark Silicon!
- **Hardware accelerate**: moving computationally intense portions of software into more efficient dedicated hardware
- **Change state**: put unused cache and memory into a drowsy state where it retains contents but consumes less power by reduced voltages

Reducing Power and energy

Observations

- A significant amount of energy goes into a single read from DRAM so memory hierarchy is key!
- Adds are significantly more expensive than multiplies
- Floating point is more expensive than integer computation

Other performance Metrics

- Million of Instructions Per Second (MIPS): depends on how it is evaluated -> Native MIPS, Peak MIPS, and Relative MIPS.

$$Instruction\ Per\ Second\ (IPS) = \frac{Instruction\ Count}{Execution\ Time}$$

$$Native\ MIPS = \frac{Instruction\ Count}{Execution\ Time \times 10^6}$$

Instruction count in terms of millions of instructions

- Peak MIPS is obtained by choosing a sequence of instructions (i.e, an instruction mix) which could provide the maximum MIPS.

Other performance Metrics

- Relative MIPS: Estimated relative to an agreed-upon reference computer (e.g. Vax 11/780)

$$Relative\ MIPS = \frac{T_{ref}}{T_{machine\_to\_be\_rated}} \times MIPS_{ref}$$

$Tref$: Execution time of reference machine

$T_{machine\_to\_be\_rated}$: Execution time computer to be rated

- MIPS varies with
  - the ISA (i.e., the complexity of instructions)
  - the choice of instruction mix (program)

- Higher MIPS does not guarantee better performance (instruction complexity)
- Relative MIPS is useful to rate evolving designs of the same computer

Other performance Metrics

$$FLOPS = \frac{Number\ of\ Floating-point\ Operations}{Execution\ Time\ (in\ seconds)}$$

- FLOPS: is used for machines used in fields of scientific calculations.
- Timeline
  - June 2007: IBM Blue Gene/L supercomputer has a peak of 596 teraFLOPS (performs 596 trillion FLOPS.)
    - ❖ used to simulate approximately 1% of a human cerebral cortex
  - June 10, 2013, China's Tianhe-2 was ranked the world's fastest with 33.86 petaFLOPS.
  - On June 20, 2017, China‘s Sunway TaihuLight was ranked the world’s fastest with 93.01 petaflops on the Linpack benchmark (out of 125.4 peak petaflops).

Mega = 10^6 Giga = 10^9
Tera = 10^{12} Peta = 10^{15}

Other performance Metrics

- MTTF: Mean Time to Failure
  - MTTF is the length of time a device or other product is expected to last in operation
  - MTTF is one of the many ways to evaluate the reliability of pieces of hardware or other technologies

Examples

-- MTTF is (perhaps) 100,000 hours for a fan

-- MTTF is (perhaps) 1,000,000 hours for a hard disk; then the failure rate is "1/1000000" per hour

$$FailureRate = \frac{1}{M} = \frac{1}{1000000},$$

$$MTTF = \frac{1}{FailureRate} = \frac{1}{\frac{1}{1000000}} = 1{,}000{,}000(hours)$$

Other performance Metrics

- MTTF: Mean Time to Failure
  - IF modules have independent, exponentially distributed lifetimes (age of module does not affect probability of failure), the overall failure rate is the sum of failure rates of the modules

Examples:

-- If there are N hard disks, each with MTTF of M hour:

$$FailureRate = N \times \frac{1}{M}, \qquad MTTF = \frac{1}{FailureRate}(hours)$$

-- Further, a system consisting with N_1 hard disks (M_1 hour MTTF per disk), N_2 disk controller (M_2 hour MTTF per controller),, and N_3 power supply (M_3 hour MTTF per supply)

$$FailureRate = N_1 \times \frac{1}{M_1} + N_2 \times \frac{1}{M_2} + N_3 \times \frac{1}{M_3}$$

$$MTTF = \frac{1}{FailureRate} \ (hours)$$

Other performance Metrics

- MTTF: Mean Time to Failure
  - IF modules have independent, exponentially distributed lifetimes (age of module does not affect probability of failure), the overall failure rate is the sum of failure rates of the modules

Examples:

-- Assuming a system consisting of N disks (M hour MTTF per disk), and the system is considered to fail if X disks fail. Thus:

$$FailureRate = N \times \frac{1}{M} / X \qquad MTTF = \frac{1}{FailureRate}(hours)$$

Quantitative Principles of Computer Design

- Take Advantage of Parallelism

  - Three examples of the use of parallelism are given in Chapter 1.9.

- Principle of Locality

  - Two different types of locality (**Temporal** locality and **Spatial** Locality) will be applied in Chapter 2.

- Focus on the Common Case

  - The most important and pervasive principle of computer design is to focus on the common case

- Amdahl's Law

  - The performance gain that can be obtained by improving some portion of a computer can be calculated using Amdahl's law.

- The Processor Performance Equation

PUTTING ALL TOGETHER: PERFORMANCE, PRICE, POWER

- Three Dell PowerEdge servers being measured and their prices (Aug. 2010

	System 1		System 2		System 3	
Component		**Cost (% Cost)**		**Cost (% Cost)**		**Cost (% Cost)**
Base server	PowerEdge R710	$653 (7%)	PowerEdge R815	$1437 (15%)	PowerEdge R815	$1437 (11%)
Power supply	570 W		1100 W		1100 W	
Processor	Xeon X5670	$3738 (40%)	Opteron 6174	$2679 (29%)	Opteron 6174	$5358 (42%)
Clock rate	2.93 GHz		2.20 GHz		2.20 GHz	
Total cores	12		24		48	
Sockets	2		2		4	
Cores/socket	6		12		12	
DRAM	12 GB	$484 (5%)	16 GB	$693 (7%)	32 GB	$1386 (11%)
Ethernet Inter.	Dual 1-Gbit	$199 (2%)	Dual 1-Gbit	$199 (2%)	Dual 1-Gbit	$199 (2%)
Disk	50 GB SSD	$1279 (14%)	50 GB SSD	$1279 (14%)	50 GB SSD	$1279 (10%)
Windows OS		$2999 (32%)		$2999 (33%)		$2999 (24%)
Total		$9352 (100%)		$9286 (100%)		$12,658 (100%)
Max ssj_ops	910,978		926,676		1,840,450	
Max ssj_ops/$	97		100		145	

What did we Learn So far

- Metrics & Enhancement techniques
  - ✓ Performance
  - ✓ Power/energy
  - ✓ Reliability
- Computer Design Principals
  - ✓ Take advantages of parallelism: Instruction-, Data- and Thread-level
  - ✓ Principle of locality: cache; temporal and spacial locality
  - ✓ Focus on the common case
  - ✓ Amdahl's Law
  - ✓ The processor performance equation

In-class Quiz

- IF modules have independent, exponentially distributed lifetimes (age of module does not affect probability of failure), the overall failure rate is the sum of failure rates of the modules

Calculate:

-- The FIT and MTTF of a system consisting of 1- disks (1M hour MTTF per disk), 1 disk controller (0.5M hour MTTF per controller), and 1 power supply (0.2 hour MTTF per supply).

$$FailureRate = 10\times\frac{1}{1000000}+\frac{1}{500000}+\frac{1}{200000}$$
$$=\frac{10+2+5}{1000000}=17\times10^{-6},$$

$$MTTF=\frac{1}{17\times10^{-6}}\approx 59{,}000(hours)$$

In-class Quiz

- IF modules have independent, exponentially distributed lifetimes (age of module does not affect probability of failure), the overall failure rate is the sum of failure rates of the modules

Calculate:

-- Assuming a system consisting of 10 disks (1M hour MTTF per disk), and the system is considered to fail if 2 disks fail. Calculate the MTTF.

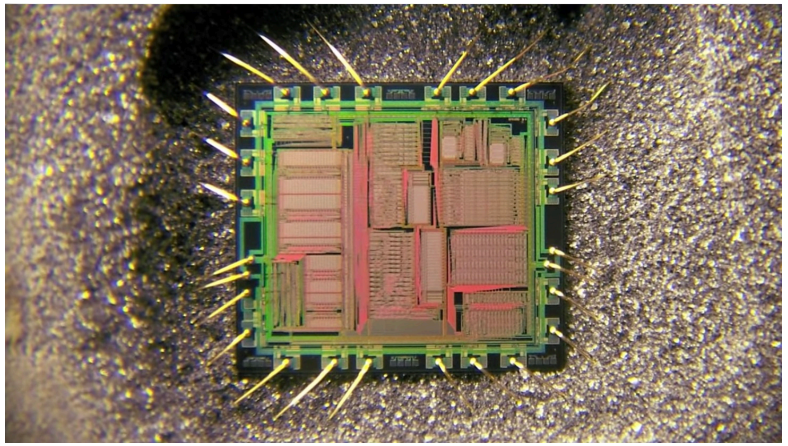
$$FailureRate = 10 \times \frac{1}{1000000} / 2$$
$$= 5 \times 10^{-6},$$

$$MTTF = \frac{1}{5 \times 10^{-6}} \approx 200{,}000(hours)$$

Power dissipation in processors, power metrics, and low-power design techniques

The Evolution of Processing Power

- The Evolution Of CPU Processing Power
  - The Mechanics Of A CPU

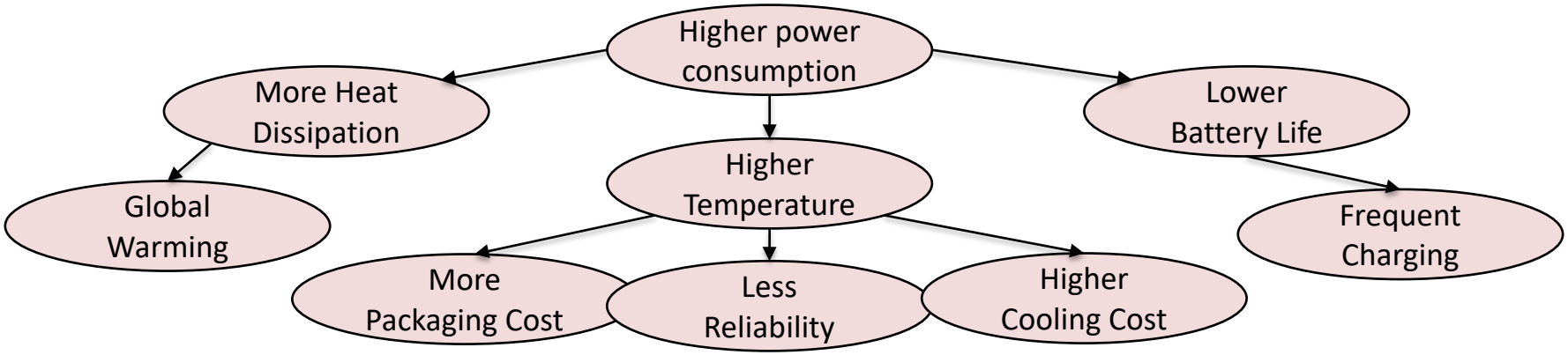


Click Figure

- The Evolution Of CPU Processing Power
  - Rise Of The x86

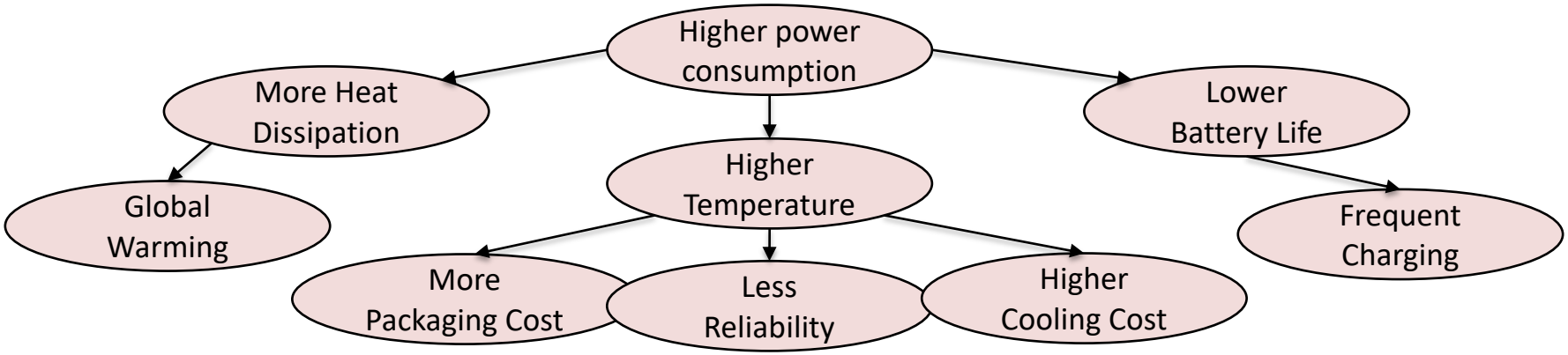
Click Figure

Need for Reducing Power



- More power dissipation makes the device unreliable
  - Power → temperature rise → temperature induced effects in device functionalities.
  - More computation-intensive applications in portable devices with growing computing power.

Need for Reducing Power



- Increasing number of battery operated devices like hand phone and tablets.
- As energy consumption increases battery life decreases.

High- performance with less power consumption is required for all kinds of computers (just not portable devices)

Power Dissipation in A Processor

- Where and How?
  - Power is dissipated in *Logic*, in *Memory*, and *Interconnections*
  - Power dissipation in *Logic Devices*
    - Dynamic power dissipated only when computation is performed
    - Static (Leakage) power is due to leakage current, and dissipated whenever system is powered-on even if no computation is done
  - Power dissipation in *Memory Devices*
    - Dynamic power is dissipation in memory occurs during read/write
    - Static (Leakage) power dissipation is identical to that in logical devices
  - Power in *Interconnect* : is the dynamic power consumption due to interconnect capacitance switching
    - Signaling interconnects convey intermediate results during computation
    - Clock distribution network carry the clock pulses

Power Dissipation in Processor

- Power dissipation
  - Dynamic power: ($Pdyn$) dissipated only when processor executes instruction.
    - It increases with operating voltage (V) and clock frequency (f).
    - The faster we compute, more is the dynamic power.
  - Static (leakage) power: (Pst) is due to the leakage current, and dissipated whenever the system is powered-on even if no computation is done.
    - It is independent of clock frequency.
    - It increases with temperature of the processor.

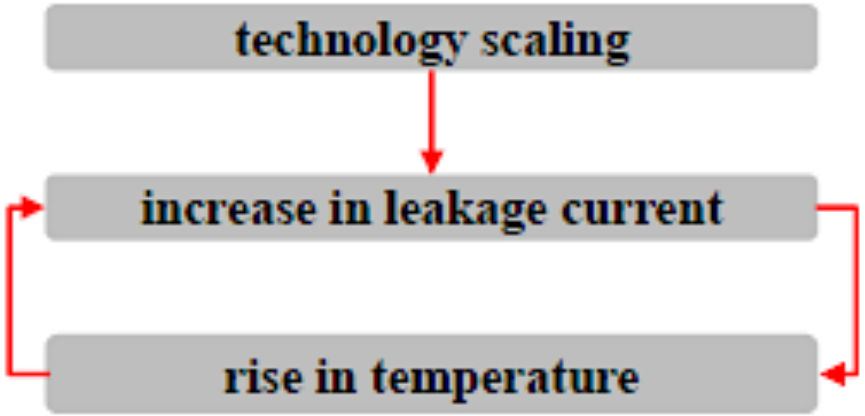
Power Dissipation in Processor

- Dynamic Power Consumption, $\boldsymbol{P_{dyn} = ACV^2f}$
  - C: total load capacitance in the circuit
  - f: clock frequency
  - V : operating voltage (also called V_{dd})
  - A: switching activity factor: the fraction of transistors switch during a clock cycle (in average).

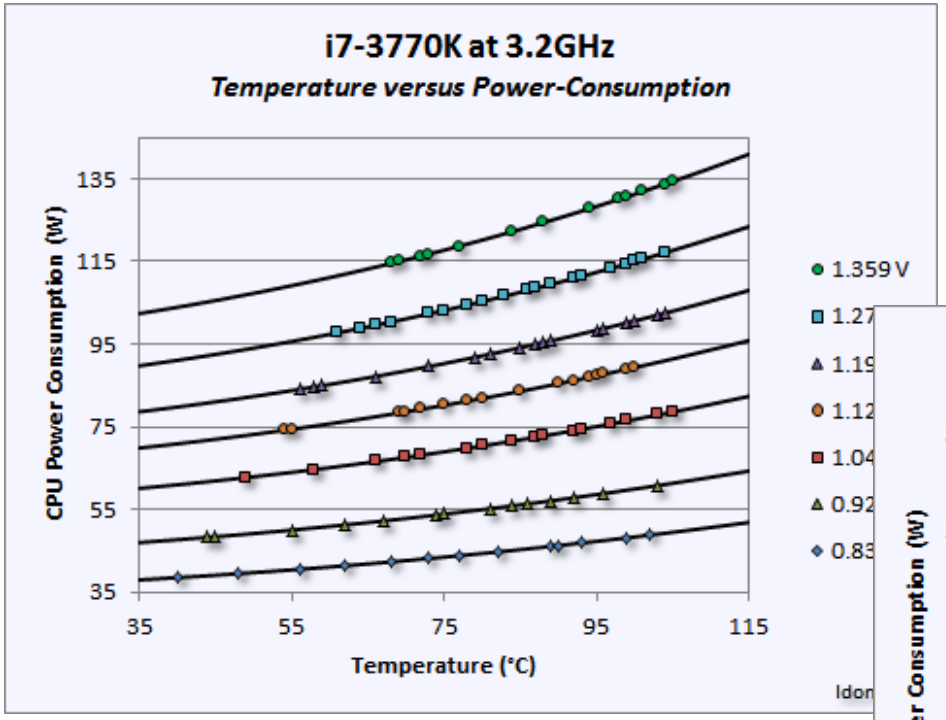
- Dynamic Power Dissipation Behaviour
  - $Pdyn \propto V^2$: Reduction of voltage reduces power consumption significantly
  - $\propto f$: Reduction of frequency reduces $Pdyn$, but degrade performance.
  - $\propto A$: Switching activity can be reduced by turning-off the unused resources/components.
  - $\propto C$: Load capacitance increases when we use more logic circuits, deeper data path. Can be reduced by inserting buffers.

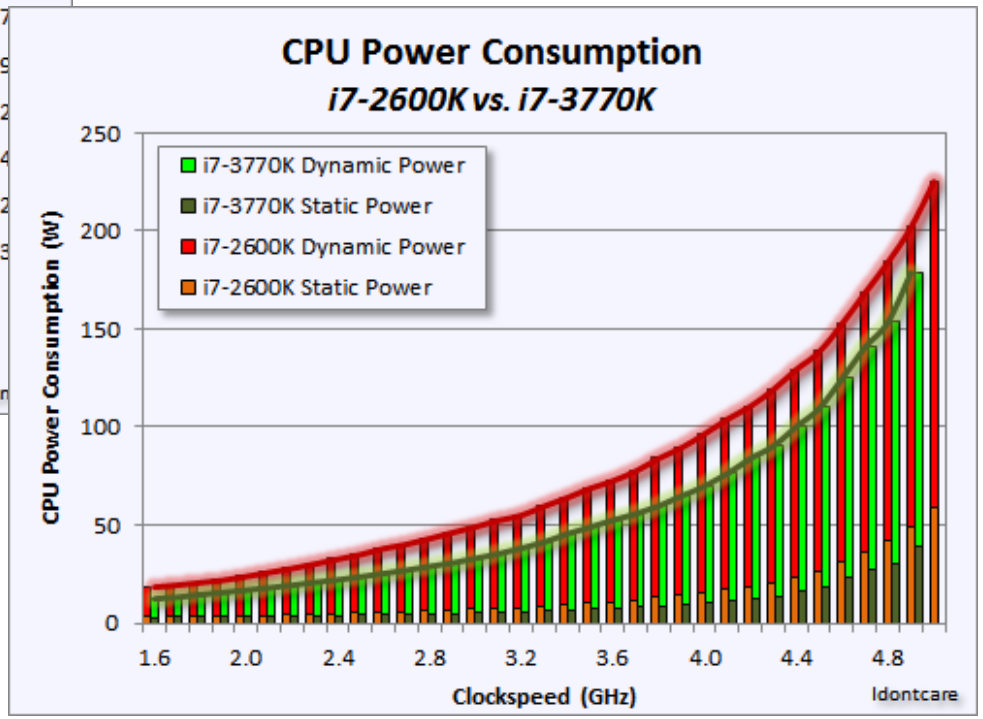
Power Dissipation in Processor

- Static Power Consumption, $\boldsymbol{P_{st} = VIleak}$
  - $Ileak$: Leakage current
  - V : operating voltage (also called V_{dd})
- Static Power Dissipation Behaviour
  - Leakage current increases cumulatively with temperature.
  - Cumulative Increase of Leakage Current with Increase of Temperature



POWER TEMPERATURE TRENDS





Source: Anandtech

Total Power Dissipation in Processor

- Total power consumption = $ACV^2f + V\,Ileak$
  - Both dynamic and static power are reduced by voltage reduction

- Maximum operating frequency $f_{max} \propto [V - V_{th}]/V$,
  - V_{th} is called the threshold voltage, the gate-source voltage at which the transistor just starts conducting
  - if V_{th} is small compared to V, maximum usable frequency $f_{max} \propto V$

What are the best methods for total power reduction?

Reducing Power Consumption

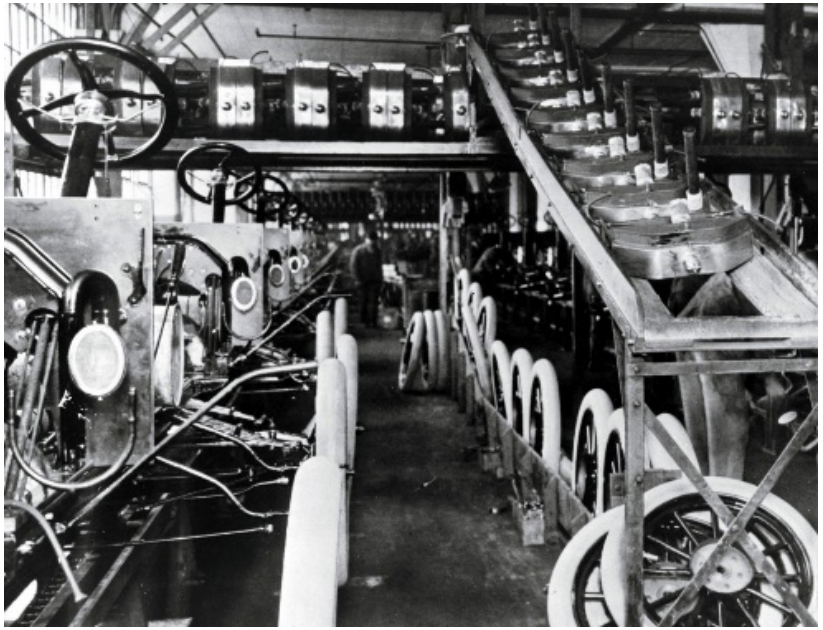
Voltage and Frequency Scaling

- Total $P_{dyn} = ACV^2 f, P_{dyn} \propto V^2$, Voltage reduction can result in considerable saving of power
- Can we reduce power consumption by reducing only frequency? Yes
- Can we reduce energy consumption by reducing only frequency? No
- We can reduce power consumption by reducing only frequency, but with the same degradation of performance.
- Energy consumption remains the same even as reduction in clock frequency increases the clock period.

$$(E = \int_{t=0}^{T} P_{avg}\, dt)$$

- Reduction of energy/power consumption by Component design: Power gating, Clock gating, Reduce data movement, number of memory access, and register transfer

What did we Learn So far



- Power Consumption
  - ✓ Contributors
  - ✓ Power/Energy consumption reduction techniques